Bachelor project - Aegis Digital Voter List

Nikolaj Aaes and Nicolai Skovvart.
IT University of Copenhagen.
Supervisor: Joseph Kiniry

May 22, 2012

Abstract

Securing modern e-voting systems is a very challenging task. This paper describes an attempt to implement a secure digital system that could assist the current Danish voter card-to-ballot exchange protocol. The current approach is paper based and we have developed a digital solution with a strong focus on securing the data using encryption. The paper also discusses the different protocols for how election data is handled, transported and who interacts with it. We identify different kinds of attacks the system could be susceptible to, and present what kinds of countermeasures we have implemented to prevent any malicious behaviour from both outside and inside adversaries.

Contents

Chapter 1

Introduction

Voting in Denmark is a paper based process prone to errors and it requires many resources. This paper describes the Aegis Digital Voter List system (Aegis DVL), designed to replace the current paper based approach of validating voters based on their voter cards with a software solution. The system handles sensitive data and needs to be resistant to malicious attacks and tampering. The paper discusses how network information is secured, how crashes are handled, how the data is distributed, and other relevant topics related to the system.

1.1 Problem definition

KMD developed a proprietary system used to generate and check voter cards. It provides little transparency and it can be hard to trust that it is secure, since the security can not be verified by the public. Is it possible to develop a transparent and secure alternative to KMD's solution?

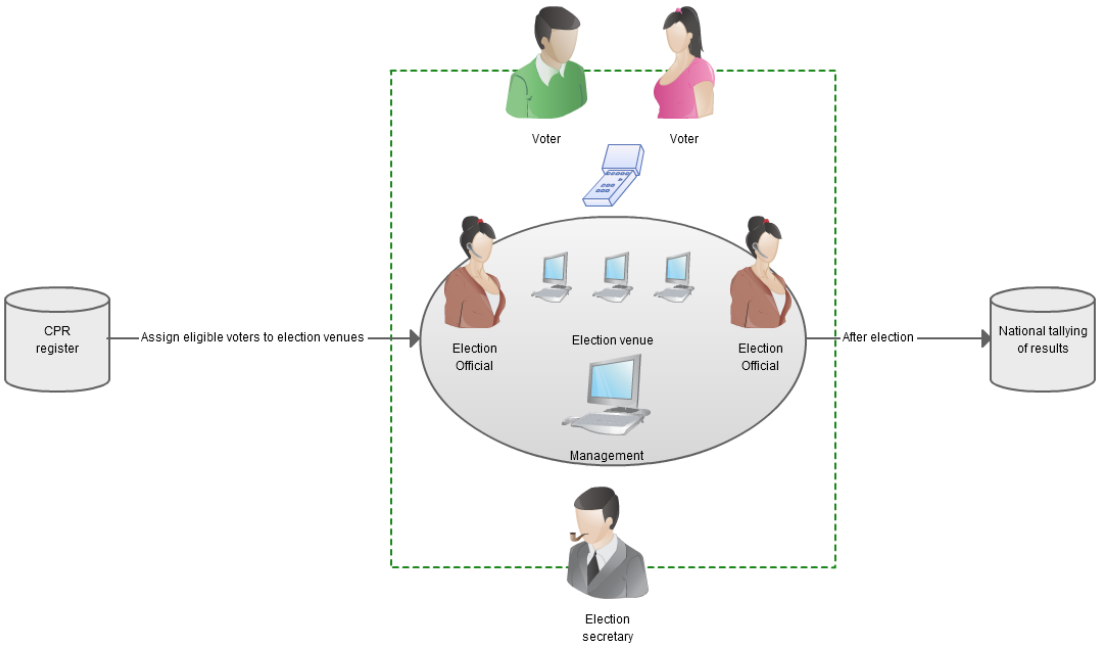
The goal of this project is to design and develop an open source replacement for the proprietary, expensive Digital Voter List system developed and supported by KMD, used to generate and check voter cards in the 2011 national elections. The system will focus on data security and consistency.

Instead of reinventing the process, we have examined the KMD system and used some of the concepts. We are not building on top of the KMD system but rather investigating other ways to handle the same problems, both regarding design and implementation. A user of KMD's system should ideally be able to sit down and use the Aegis DVL system right away.

Chapter 2

Scope

The system is responsible for the exchange of voter cards to ballots, and not the actual votes. There is only one entry point in the form of the import of voter data and one exit point when the data is exported again. An election secretary is responsible for the election venue and election officials are responsible for handing out ballots to the eligible voters.



This paper covers the following topics:

- A discussion of the design of the Aegis DVL system.
- A discussion of what data is vulnerable and should be protected, and how the security is obtained.
- A description of how synchronization and distribution of data is implemented in the Aegis DVL system and a brief discussion about the alternatives.

- A description and discussion about the security measures taken to ensure that the voting data is protected and can not be tampered with.
- A user manual describing the common usage of the Aegis DVL system.
- An overview of the testing strategy and the results.
- Notes for any future developers of this, or a similar system.

Several other topics are not included in this paper:

- No usability analysis of the user interface has been performed. It is purely for demonstration purposes, and while containing the appropriate functionality, the aesthetics was not a priority.
- This solution does not cover what happens before and after the election. This includes, but is not limited to, the partitioning of data, the printing and sending of voter cards, the storage of the machines, the collection of the data after the election has ended and the counting of votes.
- This paper does not discuss the physical transportation of voter cards, machines, USB devices etc. in depth. While physical transportation is suggested several times one must consider the logistics and how the vehicle is guarded amongst other factors before implementing the solution in real life.
- The paper does not include an economical analysis concerning the Danish election protocols and how much money can be saved by using this solution instead of the existing one.
- Neither an implementation nor a discussion of letter votes is included.

Chapter 3

Assumptions

To reason about the systems and the work practices surrounding it, we have made certain assumptions:

- Both inside and outside adversaries will use any given opportunity to exploit the system.
- Adversaries have the required resources and time to carry out the attack of their choice.
- The encryption algorithms can be trusted to encrypt and decrypt data in the manner explained in the documentation in a reliable fashion.
- The algorithm chosen for generating keys can be trusted to generate matching key pairs in a reliable manner.
- Danish CPR numbers are unique.
- A single entity holds all CPR numbers and is able to partition them for the election venues.
- A single entity will receive all the voter data from all the election venues after the election has ended.
- The entity that prints voter cards and hands them out can be trusted.
- No election venue will contain more than 25 machines.
- It is unlikely for multiple machines to fail at once unless the system is being attacked.
- Each election venue will handle at most 25.000 voters during the election.

Chapter 4

Requirements and Goals

We wanted a system which was secure and user friendly. We wanted as little responsibility transferred to the election staff as possible which means that our program should be able to solve most problems without requiring attention from the user. With this in mind we devised the following requirements:

Primary requirements:

- Features
  - Must be able to register when a voting ballot has been handed out, and prevent it from happening multiple times.
  - Must be able to confirm whether a voter is eligible to be handed a ballot based on a CPR number and a voter number.
  - Must support a management machine with elevated privileges.
  - Must have a graphical user interface.
  - At least the management machine, must be able to display relevant data about the election and status of the stations.
- Code requirements
  - Unit tests must cover at least
    - 90% of the station/manager-code.
    - 90% of the code of the database-layer.
    - 90% of the code of the crypto-layer.
    - 90% of the core data-types.
  - Other tests must include
    - The scanner.
    - The printer.
    - The user interface.
    - The communication-layer.
  - Must use code contracts.
  - Must be thoroughly documented.

- The system
  - Must be able to recover from common network errors.
  - Must be able to track if a voter card has been printed for a person.
  - Must allow a voter to use any of the stations at the election place.
  - Must allow extraction of the full data set on at least the management machine, at any given time during the election.
  - Must be able to generate voter cards.
  - Must be able to scan voter cards.
  - Requires at least four machines to operate, of which, one is a management machine.
  - Requires that adding or removing a station must be approved by at least the management machine.

Secondary goals (optional):

- It should be faster to use the system than using the current paper-based model.
- The system should be able to generate a list of all the voters of the election place and whether they have voted or not and print it.
- The graphical user interface should be easy to learn and use.
- The system should support letter votes.
- Use a data flow analysis tool to reason about correctness of the data flow in the system.
- Use an analysis tool to reason about the cryptographic protocol used.

By implementing a solution that fulfills these goals we made sure we had a well tested, documented and robust system that enabled the current work practices to be carried out in a secure manner while still being conducted inside the boundaries of the law.

Ideally the unit tests should cover 100% of the code, but as some code is hard or impractical to test, like the user-interaction and some netcode, we lowered the requirements to 90% code coverage to provide some leeway.

Chapter 5

Design and Architecture

5.1 Overview

The system we have designed consists of one manager machine and at least three station machines with the ability to add more. Each of the machines will have an attached barcode scanner that enables voters to scan their voter cards. A voter can type his CPR number into the system and scan his voter card which makes the system check if he is eligible to receive a ballot. If he is, an election official should hand him a ballot.

The system needs to be distributed because the data needs to be shared between the machines. For a discussion on how this is achieved, see section 6 Data. The sharing itself is done through the local network and this could potentially be a security concern. We require that users of the system makes sure they are connected to a closed, wired network during the entire election. This is discussed further in section 8 Security.

Since the data the system is handling is personal sensitive data, encryption of the data is essential. We strove to have the data encrypted at all times to make sure that both outside and inside attacks would be as hard as possible. This applies to the databases containing the voter data and the logs as well as the data being transmitted over the network.

To use the system one must have an encrypted data set of the voters that are eligible to vote at the election venue and the encrypted key used. This data is loaded into the system on the manager machine and when it connects to a station it is distributed to that station. The manager machine generates a master password which is used to start an election, end an election, mark a voter as having received a ballot with his CPR number only, and access the log database.

When the manager machine has connected to the desired stations, it can start the election. When this is done all the machines switch to a screen where it is possible to enter a voter number and a CPR number. It is also possible for the manager machine to remove or add additional stations on this screen. When a voter enters his voter number and CPR number and pushes the "Færdig" button, the system checks whether he is eligible for a ballot or not. If a voter has lost his voter card, the election secretary can mark a voter as having received a ballot, using just his CPR number and the election venue master password.

When the election ends all the stations close their application and the manager machine can

export the data to a file location. The exported data is still encrypted and can only be decrypted by the holder of the initial decryption key, that was generated with the voter data encryption key.

As a rule of thumb, the system was designed to shut down the election if the suspicion of an attack is raised. Since no guarantees can be given about a data set that was potentially a victim of an attack, the risk is too high to continue the election. If the manager machine becomes unreachable, an election for a new manager will start and an active station will be promoted to be the new manager when it ends. This promotion can also be done through the manager's user interface. If a station becomes unreachable it will be removed from the list of active machines the other machines know.

5.2 Design

Choosing the right security mechanisms was a major part of our design decisions and we approached this using the twelve principles presented in Applied information security: A hands-on approach [1] which are discussed in section 8 Security.

We have used the BON design language [33] in our design process to get a complete overview of our application before producing any code. We used code contracts [34] to make sure the application behaved as expected as dictated by the Design by Contract principle [35].

To improve the modularity of the application we provided interfaces for all the major classes except for Station. This makes for easy replacement of parts of the program which might become needed later on. We used the Mediator pattern [43] when we implemented the user interface since we wanted it to be easily replaceable with any user interface. The only requirement would be to implement the IDvlUi interface to make sure the back-end of the system could communicate with the user interface.

As for the messages sent from machine to machine we used the Command pattern [45] which provided us with an easy way to encapsulate data and instruct the target machine what to do with it.

5.3 The main classes

To provide an overview of the classes in the application we have created a class diagram which can be found in Appendix 17.2 Class Diagrams, along with descriptions of the major classes in the system:

5.3.1 Station

The Station class is the large back-end class that contains the core functionality for the station and manager machines. While a station machine and a manager machine have semantically different meanings, in the code, the Station class contains functionality for both, since a manager machine is merely a station machine with elevated rights and responsibilities. As such we have compiled a list of functionality the Station class contains and whether it is used by the manager machine or a station machine:

- Station

  - Start election for new manager.
  - Request a ballot.

- Manager
  - Add/remove stations.
  - Transfer manager-status to station.
  - Check status of stations.
  - Start election.
  - End election.
  - Manually mark selected voter as being handed a ballot (in case they lost their voter card).

5.3.2 Crypto

The Crypto class is responsible for all encryption and decryption related actions. It can encrypt and decrypt with both symmetric keys and asymmetric key pairs. It is also used to generate the master password and the required key pairs. If the encryption and decryption algorithms need to change, a new Crypto class can be constructed and used as long as it implements ICrypto.

5.3.3 Communicator

The Communicator class is responsible for the network communication between machines. It both sends and listens for commands, and executes each command as it is received. If the network protocol needs to change, a new Communicator class can be constructed and used as long as it implements ICommunicator.

5.3.4 SqLiteDatabase

The SqLiteDatabase class facilitates all queries to the database. This system uses an SQLite database, but it can easily be changed and the alternatives are discussed in section 7.1 Database management system (DBMS). If the DBMS needs to be changed or one wants to change to a different kind of data storage, a new database class can be constructed and used as long as it implements IDatabase.

5.3.5 Logger

The Logger class is responsible for all log entries and exporting the log. Whenever an important event in the system occurs, the Logger class sees to that it is logged in the right place with the right encryption. No logging framework is used by our logging class, but if one wanted to add a framework or change the way the logs are stored, a new Logger class can be constructed and used as long as it implements ILogger.

5.3.6 UiHandler

The UiHandler is responsible for all user interface related communication. Every time the user interface wants to use methods from the station and the other way around, it results in a call to the UiHandler. If the user interface needs to be replaced a new UiHandler class can be constructed and used as long as it implements IDvlUi.

5.4 Generating voter cards

One of the requirements for the system was the generation and printing of voter cards. To accommodate this we have added a PDFGenerator project written by Kåre Sylow Pedersen as a part of the Digital Voter Registration System [29]. The code can generate voter cards and lists of voters and requires code contracts to be installed. This is not part of the user interface, because generating and printing voter cards takes place before the election starts, and will not be printed at the election venues. Every time a voter card is printed, it should be saved in an appropriate database. There is no reason for this data to be distributed to the election venues since it is not used in the system, but the entity printing the voter cards might have a use for it.

It is recommended to use a scanner with our current user interface since the generated voter cards have barcodes associated with their voter number. We tested the system with a Symbol HotShot LS2106 barcode scanner which essentially fires keyboard events when it scans. As long as the correct text box has focus the scanning works as intended. This scanner was produced in may 2000 and uses a PS/2 keyboard input.

5.5 Contract coverage

We have used code contracts in our system to ensure that our code will always function as long as the contracts are respected. It also makes debugging easier as a failed precondition will stop execution immediately instead of passing potentially bad parameters to other methods. The use of preconditions also allow us to ignore a lot of exception-throwing code as errors can be made impossible as long as preconditions are abided by. The contracts cover the following of our code.

Contract coverage results

Domain	**Count**
Total amount of methods	158
Methods covered by contracts	93
Lines of contract-code	189
Lines of non-trivial contract-code	39
Class-invariants	9

It is worth noting that a lot of the methods that are not covered are auto-property getters that are unable to guarantee anything. The majority of the contracts are trivial requires-not-null checks or ensures-not-null checks. Some of the more interesting contracts requires that stations are (or are not) currently listening to TCP requests, or requires that the machine is currently the manager.

Chapter 6

Data

This system handles a lot of data transactions and most of this data is personal and sensitive. People do not want everyone knowing their CPR numbers and whether they have voted or not. Before an election can start, each election venue needs a list of voters that should be able to hand in their voter cards in exchange for a ballot and vote at their specific location. Initially all this information is stored in a single location and needs to be partitioned for each election venue. This partitioning will most likely be based on the addresses of the voters, but in this paper we do not discuss how this partitioning should be conducted.

After the partitioning, the different fragments must be transported to the election venues. This can happen in a few different ways:

- Use the Internet to transmit the data.
- Use a messenger service to transport it via a portable medium (USB device, CD etc.).
- Use your own messenger to transport it via a portable medium.

We strongly recommend the "Use your own messenger to transport it via a portable medium"-approach to reduce the attack surface for adversaries and to gain more control of the transportation. The transportation should preferably be guarded, but the financial costs of this might exceed the benefits.

6.1 Receiving and distributing data

When the partitioned data arrives at the election venue it needs to be distributed to all the machines in the election. To make it easier for the person who needs to set up the machines at the election venue it is assumed that there is a single point in the closed network that receives the collection of eligible voters. This makes for a few possible solutions for receiving the data:

- A manager machine receives the data and distribute it to the other machines.
- A station machine receives the data and distribute it to the other machines.
- Either a manager- or a station machine can receive the data and distribute it.

Alternatively the data could be distributed manually via a portable medium, but this is unnecessarily cumbersome. We have chosen that the manager machine receives the data and

distributes it. Since the manager is the machine managing the stations, it makes sense to have this machine join the task of receiving and distributing data with the task of connecting to all the stations.

The data can be distributed among the machines in several different ways each with its own advantages and disadvantages.

Every machine has the full data set all the time.
This solution has the advantage of being the most robust, because the data is not lost if a machine crashes, since all the other machines will have a full backup of all the data. The disadvantages are that the network traffic required to makes sure that the data set is up to date on all the machines is quite high compared to the other solutions. Also, if an adversary was to gain access to any machine he would have access to the full data set which leaves him with a larger attack surface.

Every station has a partition of the data set and the management machine has either no data set, the full data set or a backup partition based on some criteria.
This solution uses less network traffic since it only needs to synchronize the station with the relevant part of the data. Also this solution leaves less options for adversaries to gain access to the full data set since each machine only has a partition. The disadvantages is that the solution is very prone to adversaries that seek to destroy the election. If even a single machine crashes, its entire data set is lost. This can be circumvented by having a backup of the full data set stored on the manager machine which will increase network traffic, but provide a full data set which increases the attack surface.

Every station has two or more partitions of the data set, one partition belonging to the station itself and one or more backups of the other stations. The management machine can have data sets like in the second solution.
This solution improves on the previous solution by having a more robust design. In this solution a machine can crash without the loss of data since a backup is always kept on another machine. This increases the network traffic, but leaves the full data set partitioned making it harder for adversaries to obtain it.

The management machine has the full data set and the stations contain no data.
This solution focuses on storing as little data as possible on the stations. Since the stations are the most vulnerable machines, as they are handled by the voters, they contain no data at all. This is somewhat network traffic intensive for the manager, compared to the other solutions, since every update is sent to the manager who then updates the database. It is also quite a dangerous solution since the manager machine becomes a single point of failure. If it crashes the entire election data is lost. Against adversaries this is both advantageous and disadvantageous since the stations have no data that can be obtained, but the manager machine has the full data set. If the adversary is aware that the data is located on the manager machine only, he has no need to attack the stations.

A separate database is located in the election venue and the management machine takes the role as a proxy to facilitate communication between stations and the database.
This solution is quite similar to the previous solution, but the data is now moved to a separate machine. This is an advantage because the manager machine facilitates other features and is therefore more prone to errors and attacks than a separate machine which no one interacts with. The disadvantage is an increase in network traffic since the manager

now has to forward all requests and answers from the separate database. This solution still has a single point of failure which, from a distributed systems viewpoint, is a serious disadvantage.

We chose to use the first solution for its robustness. We realized that we needed to focus on making each machine as secure as possible since they all contain the full data set, but being able to recover from the crash of any machine is a desirable property. While this solution is traffic intensive, we do not sacrifice any robustness and in a real world scenario each election place has at most 25 machines in total, which makes the traffic almost unnoticeable. The system might not scale in an ideal manner, but the security aspect takes priority over performance.

Chapter 7

Synchronization and Broadcasting

Since we chose a robust solution where every machine has the entire data set at all times, we need a way to synchronize all the machines to make sure that all the data sets are up to date if any of them should crash. There are several ways this can be done:

- **Request synchronize** - A station requests the manager machine to synchronize all the other machines with a certain update set.
- **Broadcast** - A station broadcasts an update set to all other machines.
- **Epidemic** - A station utilizes an epidemic protocol to update all other machines.

This synchronization can be initialized at different times during the election:

- **On action** - After every action (a voter scans a voter card) on a station, a synchronization is initialized.
- **Interval** - At a certain time interval a system wide synchronization is initialized.
- **Key-points** - At certain key points (eg. after 100 voter cards have been scanned) a system wide synchronization is initialized.

We have chosen a combination of "Request Synchronize" and "On action". By using the manager as a mediator when an update is to be propagated to the machines in the network, we obtain a simpler communication channel which is easier to reason about and test. We chose "On action" updates because we want the updates to happen every time a voter has been handed a ballot, to ensure, that if a machine crashes its data is not lost. We realize that this generates a large amount of messages, but it satisfies our condition, that every machine must have the full data set all the time as described in section 6.1 Receiving and distributing data.

Once again there are several ways we can do this:

Our own algorithm - with this approach an update message is sent from a station to the manager every time a voter requests a ballot. The manager checks its own database and if the voter is eligible for a ballot, it sends a message to every station other than the initial one telling them to update their database. Lastly the manager sends an update (and confirmation) to the initial station which then hands out the ballot. If the initial station becomes unavailable (i.e. crashes) before it can receive a confirmation, the manager

sends out a revoke command to every other machine telling them that the ballot has not been handed out and that their database should reflect that. It is important that the manager sends the update messages at the same time, because the system can not handle a situation where the manager crashes halfway through updating the stations. That leaves some stations with one ballot status and some with another and no manager to confirm which one is correct. If a station is unreachable when an update message is sent, it is removed from the manager's and the active stations' list of connected stations.

The ChandyLamport algorithm (Snapshot algorithm) [2] [24] - with this approach an observer process initiates the algorithm to gather a global snapshot of the system. If we were to use this algorithm it would have to be modified since we wanted updates to be communicated to other machines when a ballot has been handed out, and this algorithm only updates the initiator. The most significant problem however, is the fact that the entire state of each machine would be sent over the network. This could potentially be thousands of entries which is unnecessary for our purposes.

NSync [26] - NSync would be a good choice if we wanted to have several updates at a time on each machine. It works by sending metadata on what changes needs to be made, resolves conflicts and afterwards sends the necessary data for the changes to happen. It would not be fit for our purpose since we want to send one update at a time and because that conflicts in the data sets, is a reason to suspect that a machine has been compromised in our system.

To provide better insight into how our algorithm is implemented, the following pseudo code is supplied:

Algorithm 1 Our synchronization algorithm - Station side

1: $VoterNumber \leftarrow$ Scanned VoterNumber
2: $CPR \leftarrow$ Typed CPR
3: $Check \leftarrow CheckOwnDatabase(VoterNumber, CPR)$ {returns false if the voter does not exist or has already received a ballot}
4: **if** $!Check$ **then**
5: $InformVoter()$ {inform the voter that he does not exist or has already received a ballot}
6: **else**
7: $Manager.RequestBallot(VoterNumber, CPR)$ {sends a command to the manager with the request}
8: **end if**

Algorithm 2 Our synchronization algorithm - Manager side (RequestBallot)

1: $VoterNumber \leftarrow$ Scanned Voter Number
2: $CPR \leftarrow$ Typed CPR
3: $Check \leftarrow CheckOwnDatabase(VoterNumber, CPR)$ {returns false if the voter does not exist or has already received a ballot}
4: $UpdateOtherStations(VoterNumber, CPR)$
5: **if** $IsActive$(Sender) **then**
6: **if** $Check$ **then**
7: UpdateSender(true) {sends a command to the sender telling it to update its database and tell the voter he can receive a ballot}
8: **else**
9: UpdateSender(false) {sends a command to the sender telling it not to update its database and tell the voter he can not receive a ballot}
10: **end if**
11: **else**
12: $RevokeBallot(VoterNumberCPR)$ {revokes the ballot status on all the other stations}
13: **end if**

To ensure that our algorithm works as expected, we used the model checking tool UPPAAL [28]. By using this tool we were able to verify that our synchronization algorithm updates all the machines when a ballot is handed out, and that each voter can only be handed one ballot. Screenshots from the verification can be found in appendix 17.4 UPPAAL.

We considered the fact that if an election venue has a large amount of stations, the manager might get a message implosion where too many messages are to be handled at the same time. Implementing a queue system on the manager side of the communication layer should be sufficient to handle the inbound messages. If this was a greater concern Schooler's suppression algorithm [25] would be a viable way to avoid this problem.

7.1 Database management system

To manage the data on each machine, our system uses a database management system (DBMS). We have made it easy to exchange this DBMS with another one by defining an interface for the database layer. If one were to exchange the current DBMS the properties of the new DBMS should be considered. Some desirable properties are:

- ACID (atomicity, consistency, isolation, durability) transactions either through locking or multi-versioning.
- Security layer for encryption.
- Scalability.
- Logging framework.

One might consider a DBMS with a distributed protocol to handle consistency over a network, but we have chosen one without it to get a greater degree of control on how the data in synchronized between the machines. If a DBMS with a distributed protocol is chosen, it needs to have eventual consistency within a time frame (depending on the amount of stations) to make sure the election machines are consistent between every ballot handed out. We suggest an open source system

for several reasons; an open source DBMS project could be forked to fulfill possible future requirements, it would be possible to have a peer review of the crypto layer and other security aspects, and it would also be consistent with our own open source project. There is nothing preventing the use of a proprietary system, though.

We have provided a list of some of the database management systems that could be usable and what properties they fulfill.

Database Management Systems

Name	Developer	Open source	Crypto layer	ACID	Maintained
REDIS[23]	Salvatore San Fillippo	Yes	No	No	Yes
MongoDB[21]	10gen	Yes	No	No	Yes
CouchDB[22]	Apache Software Foundation	Yes	No	Yes	Yes
MySQL[13]	MySQL	Yes	Yes	Yes	Yes
PostgreSQL[15]	PostgreSQL	Yes	Yes	Yes	Yes
SQLite[16]	SQLite	Yes	Yes	Yes	Yes
DBMS_crypto[10]	Oracle	No	Yes	Yes	Yes
MSSQL[11]	Microsoft	No	Yes	Yes	Yes
Sybase ASE[19]	Sybase	No	Yes	Yes	Yes
DB2[20]	IBM	No	Yes	Yes	Yes
Firebird[18]	Firebird	Yes	No	Yes	Yes
Microsoft Access[14]	Microsoft	No	Yes	No	Yes

We have decided to implement the database using SQLite [17]. SQLite is a "*software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine*"[17]. We decided to use this DBMS as it fulfills all the desired properties, it was fast to install and implement, and it did not require the use of external systems. To interact with the database, we use the ADO.NET Entity Framework [47].

Chapter 8

Security

Security is an essential part of every system in the domain of elections. Making sure that the election can not be tampered with, is of the highest priority because the information could potentially have consequences for a lot of people. We approached this using the twelve principles presented in Applied information security: A hands-on approach [1]:

1. **Simplicity - simpler security mechanisms are easier to understand and maintain.**
   We designed a system that introduces as few new concepts as possible, so users of the current paper-based solution should find the application easy to use.

2. **Open design - a system should not depend on the secrecy of its protection mechanisms.**
   Our system is open source and everyone can examine the code. If the security was dependant on the secrecy of the mechanisms it would effectively have no security at all. We have designed mechanisms that depend on the secrecy of generated keys and not knowledge of the mechanisms themselves.

3. **Compartmentalization - Organize resources into isolated groups of similar needs.**
   We have divided the code into classes corresponding to their responsibilities. We have provided interfaces for some of the more interesting classes which makes it easy to replace and maintain them.

4. **Minimum exposure - minimize the attack surface the system presents to the adversary.**
   By providing the minimum amount of opportunities for manual input from anyone and operating in a closed network we strove to minimize the attack surface as much as possible.

5. **Least privilege - any component of a system should operate using the least set of privileges necessary to complete its job.**
   We keep all data encrypted during the entire election to prevent anyone, even an insider, from tampering with the data. Decrypted data is never stored and as soon as new data enters the database it gets encrypted. By using a master password we ensure that only the appropriate members of the election staff has the privileges to perform certain actions such as marking a voter using only their CPR number.

6. **Minimum trust and maximum trustworthiness.**
   We choose to minimize the trust between the different machines. Every message sent over

the network is validated and if the message is not accepted, the election will switch to a paper based approach since the sender is regarded as compromised.

7. **Secure fail-safe defaults - the system should start and return to a secure state in the event of a failure.**
   We use several detection mechanisms to catch failures and handle them. See section 8.3 Detection and recovery.

8. **Complete mediation - access to any object must be monitored and controlled.**
   By using code contracts and rigid logging we monitor all access to the data. To control access to the system we only accept incoming net traffic in a certain format and to control the access to the data we use our database layer which can only be accessed through the application.

9. **No single point of failure - build redundant security mechanisms whenever feasible.**
   We do not want any machine to be a single point of failure, and by having all the data distributed to all the machines we can handle the crash of any machine. If the manager machine should crash the stations can elect a new manager and continue with the election.

10. **Traceability - log security-relevant system events.**
    We store logs locally on each machine encrypted with the master password ensuring that the log is accessible even after a system crash.

11. **Generating secrets - maximise the entropy of secrets.**
    All of our generated secrets are created using Bouncy Castle's [51] SecureRandom class. Randomly generated numbers are generally too predictable and thus insecure, so a strong source of randomness is needed. We trust that the implementation by Bouncy Castle is sufficiently secure.

12. **Usability - design usable security mechanisms.**
    The system uses several different mechanisms and we have automated as many as possible. We require very little of the users, and the tasks the users have to perform are trivial.

By following these principles we got some desirable properties for our system. The next thing to consider was what kind of attacks our system could be a victim of. For this we used the STRIDE [8] threat categories and the corresponding countermeasures:

Spoofing - We use strong authentication and store all the data in an encrypted fashion.

Tampering - We use a secure communication protocol and hybrid ciphers.

Repudiation - We use logs and digital signatures to ensure this.

Information disclosure - We use strong encryption algorithms.

Denial of service - We make sure that the machines are on a closed network with no access to the Internet.

Elevation of privilege - We follow the Least Privilege principle.

The system must be connected to a closed network during the election, only potentially connecting to the outside to import partitioned voter data prior to the election or upload exported voter data afterwards. To ensure that the network is actually closed the connection must always

be wired and not wireless. The unused port in the switch/router must be obstructed thus preventing adversaries from plugging their own machines in and accessing the network. Ideally the switch/router is in the same location as the manager machine to make monitoring both of them at the same time convenient.

To avoid that voters or election staff accidentally close the application during the election, we have disabled the red x in the upper right corner of the application. As an additional security layer we would have liked to implement the application in such a way that it would run as a service and require administrator rights to close it to further increase the security.

The master password and the decryption key to the data set are each entrusted to a single entity, which means that the two entities in question must be trusted. Ideally each of these keys would be split into several fragments and each fragment given to a different entity, preferably with different stakes in the election. Only by using all the fragments at the same time would the key be usable. This would place the trust on several entities instead of a single one and make it harder for adversaries to acquire the combined key. This is not a practical solution for the master password since you would have several entities typing on the same machine each time a voter has lost his voter card, which would be cumbersome. But for the decryption key, this would be a thing to consider when decrypting the voter data.

8.1 Attack model

To identify and assess threats to the system we created attack trees [6] using the notation described by Moore et.al. [5] with added notation for reusing attack patterns inside the attack trees to provide smaller and clearer attack trees. The full attack trees can be found in appendix 17.5 Attack Trees, where the additional notation is also described. Constructing attack trees is a method to identify different kinds of attacks against a system, consider the likelihood and resources required of each attack and manage the risks. The weakness of this approach is that it relies on the creator to consider all the different kinds of attacks and predict the correct probabilities and resources. The detail and depth of the attack tree is also decided by the creator and important information might be omitted. Since this is a paper primarily concerning software, we have chosen to focus our attacks on how one could destroy or tamper with an election via our software. By identifying the possibilities of potential adversaries we produced countermeasures and implemented a more secure solution.

The outcome of constructing attacks trees was knowledge of where to focus our efforts when designing the security of our software and we arrived at several conclusions:

- The portable medium used to transport the data from the partitioning venue to the election venue should have a protection mechanism to prevent tampering with the data prior to and after the election. This could potentially be solved by having the data obfuscated or signed, and the deobfuscation password only being exchanged securely when the data is at the election venue.
- Access to the machines used for the election must be very limited before and after the election. It should be impossible for unauthorized personnel to gain physical access to the machines prior to and after the election.
- The process of identifying voters that lost their voter cards must be very thorough before handing them a ballot to prevent impersonation.

- Being connected to the Internet can be a huge threat and should be avoided as much as possible.
- Data should be checked every time it travels from one machine to another to prevent using corrupt or invalid data.
- The hardware facilitating the network should be under observation during the election to prevent unintended machines from connecting to the network.
- Connecting to the network of machines running the software should require authentication to make it harder for adversaries to gain access to the network.
- The less decryption that takes place during the election the better. Ideally each machine should only be able to see the data it needs and nothing more thus following the Least Privilege Principle [1].
- The election personnel should consist of trusted individuals. Even though the software will protect against insider attacks, they are still one of the greatest potential threats.
- The generation of the keys used to encrypt the initial data set and decrypt the final data set should be conducted in a safe location since the acquisition of these could compromise the entire election.
- The machines used in the election should be dedicated only for the election. This should prevent the machines from being compromised prior to the election. Alternatively the machines could be reset to factory standards instead of being dedicated.

In the attack trees the attack pattern "Manipulate persons" is used repeatedly indicating that this is a weak point in the security structure. When in a real life environment, it is therefore important to make sure that the election staff is well protected and not likely to receive bribes. When the "Manipulate persons" attack pattern is used, it is often to gain access to a certain encryption or decryption key, or to the election venue and hardware. This is something that is available to the election staff as well, and if the adversary knew an insider, or was an insider himself, the "Manipulate persons" attack pattern would not be a necessary action for the attack to succeed. It is important to notice that the attack trees are devised from an outside adversary's point of view and many other obstacles would be removed as well if the adversary was an insider.

As an addition to our attack trees we considered using Microsoft's Threat Modeling approach [8], but found the threat rating method to not suit our needs and that the information we would have gained from using this method was already largely covered by the attack trees.

8.2 Protection

The system uses multiple layers of protection.

- Symmetric encryption of the log-database.
- Symmetric encryption of the voter data-database.
- Asymmetric encryption of the voter data - the voter number, CPR number and ballot status.
- Obfuscation of public keys during key-exchange, to prevent man-in-the-middle attacks.

- Hybrid-cipher encryption of (most) commands transmitted over the network.

The symmetric encryption of the log- and voter data-database is handled by our database implementation using SQLite, as SQLite has an optional crypto-layer. The log-database is encrypted with the master password, so no logs are lost due to system crashes since the password is not lost if a crash occurs. This does enforce a higher reliance on the integrity of the election secretary. The voter database password is randomly generated and known only by the machine. The voter data is asymmetrically encrypted before arriving at the election venue together with the public key that was used to encrypt the data set.

Every station has it own public/private key pair, and it shares the public key with all of its peers. During public-key-exchange, we need to be able to verify that the received request is actually from whom it claims. To do this, the public-key is obfuscated before being transmitted over the network, and the receiver has to type in a password that is shown on the sender's machine. The process is repeated the other way around and both machines should know each other's public keys.

After public-keys are exchanged, all messages, except the message checking if a station is reachable, switch to using hybrid-cipher encryption that automatically ensure that only the sender and the receiver understand the message.

During the election, there should be taken certain precautions outside of the system. The election should make use of the four-eye principle [46] making sure that there are at least two people monitoring every station, to reduce the chance of insider attacks and to make sure that no unauthorized personnel tampers with the hardware. The stations should not be connected to the Internet, and the machines external-input devices such as the USB-slots, CD-drives, etc. should be made unavailable. The manager machine will initially need to allow one of these options to import the data and the voter data encryption-key, but it should be made unavailable after initialization. To protect against potential errors, it would also be ideal if the machines and the router/switch ran on an uninterruptible power supply (UPS).

8.2.1 Input validation

Input validation is potentially an important subject, especially when working with SQL-databases. SQL-injections are a commonly known problem in many programs, especially in web-applications.

The input our system accepts is:

- Voter numbers and CPR numbers.
- Passwords (strings), the master-password and deobfuscation passwords used when exchanging public keys.
- Voter data to be imported during system initialization, and the key used to encrypt the data.
- Commands transmitted over the local network.

The voter numbers and CPR numbers are relevant as they are used in conjunction with the database (though they are not stored as numbers in the database). The fact that they are numeric makes it fairly simple to filter out bad input, and it can be handled by the user interface.

We also used the ADO.NET Entity Framework [47], an Object Relational Mapping-framework [48]. A framework such as this enabled us to work with type-safety, and reduces the risk of human error since it abstracts away from writing raw SQL-commands in strings.

The passwords are not used in any queries, and should not introduce any SQL-injection possibilities.

The voter data to be imported is serialized system structs, so when de-serializing them they should fail before ever reaching the system if they are not in the correct format. Currently, we have no way to ensure that the intended data set is the one reaching the election venue. This could potentially be solved by having the data obfuscated or signed, and the deobfuscation password only being exchanged securely when the data is at the election venue.

Commands are validated by the fact that almost all commands are sent securely wrapped in a CryptoCommand. The CryptoCommand checks that the sender is who it claims to be through the use of hybrid-cipher-encryption. This requires that the sender and receiver know each other, which they do not at system startup. Therefore, PublicKeyExchangeCommands are sent unencrypted, but the public key they contain is obfuscated by a randomly generated password. The password is shown on the sender's machine when received, and the receiving machine needs to type it in. The only other command that is not wrapped in a CryptoCommand is the IsAliveCommand, that is used to check if a machine is actively listening on the network port the system uses.

8.2.2 PGP, GPG and SSL

During our design phase we considered using PGP [44], GPG [49] and SSL [50] which are all technologies that concern themselves with secure communication. The main idea behind PGP and GPG is that you can not trust a sender of a normal email to actually be who he claims to be. This is solved by having public/private key encryption and signing of keys. While the public/private key encryption is an idea we also have used, the signing of keys does not benefit our system all that much. The value of a signature originates from the writer of that signature and if our system operates on a closed network the only machines who could sign the keys would be machines we essentially controls ourselves. This would mean that we simply trust our own signature which does not provide any security.

Alternatively the keys could be generated beforehand, imported along with the voter data and signed by an entity outside the system. This would require that each election venue would have knowledge about how many machines they would need to create the correct number of keys. One could also generate extra keys for each venue in case of system crashes. Another idea could be to have people sign the keys manually. If the election has a group of trusted people they could potentially visit all the election venues and sign the keys. While both these ideas are viable they introduce extra costs and extra complexity into the system and we have chosen not to implement any of them.

One of the things we used from the PGP and GPG technology was the idea of hybrid ciphers. This is an easy way to ensure data integrity and non-repudiation. A description of how we used hybrid ciphers to construct the commands in the system can be found in section 8.2.3 Cryptography.

Secure Sockets Layer (SSL) is a secure way to communicate over the TCP protocol and relies on digital certificates to authenticate machines. The main idea is that if a certificate authority trusts a machine to have a certain identity you could trust that identity is their real identity. This is done by asking the certificate authority for the encryption key to the machine in question and by using this you can establish a secure communication channel. We encounter the same problem as with the PGP and GPG solution. If we operate in a closed network the certificate authority must be in the same closed network for us to access it. We do not want any machines we do not control ourselves in our network, which means we have to control the certificate authority ourselves. This comes down to trusting the certificates we made ourselves, essentially trusting that we are trustworthy which does not provide any security.

8.2.3 Cryptography

Our cryptography is implemented using Bouncy Castle's [51] C# implementation. For asymmetric encryption, we use RSA [52]. Input byte-arrays are padded with a 1-byte to prevent data-loss. Other padding-schemes were tried such as OAEP [53] (Optimal Asymmetric Encryption Padding), but they made encrypted data incomparable which was needed for the database. We did not deem it a big problem, as all asymmetrically encrypted data should be unique. CPR numbers are unique, voter numbers are unique, and the ballot status (converted to an unsigned integer) is added together with the CPR number before being encrypted, making it unique. A ballot status added together with a CPR number is potentially not unique, but it has different meanings. RSA-keys are generated using Bouncy Castle's RsaKeyPairGenerator with 3072 bit strength. RSA claims that 1024-bit keys are likely to become crackable between 2006 and 2010 and that 2048-bit keys are sufficient until 2030. An RSA key length of 3072 bits should be used if security is required beyond 2030 [58].

For symmetric encryption we use AES [54] in CBC-mode [55] (Cipher-Block-Chaining) with PKCS7 [56] padding and initialization vectors (IVs). Keys and IVs are generated using Bouncy Castle's SecureRandom class. The generated keys use the highest strength supported by Bouncy Castle, which is 256 bit (32 bytes). The fastest supercomputer in the world would in theory require about $3.31 \cdot 10^{56}$ years to exhaust the 256-bit key space [59]. Ideally we would use CCM-mode [57] since it seems to be the best option Bouncy Castle offers, but we had some problems implementing it, and believed CBC-mode to be sufficiently secure. Even better would be CWC [57]-mode, but Bouncy Castle does not offer this. Our basis for this prioritization is taken from the Secure Programming Cookbook for C and C++ [57].

Our system uses asymmetric encryption for the voter data (all unsigned integers) and for encrypting symmetric keys.

Symmetric encryption is used to encrypt the network traffic in the CryptoCommand.

The CryptoCommand consists of:

- An IV - unencrypted.
- A symmetric key - asymmetrically encrypted with the receiver's public key, so only the receiver can decrypt it with his private key.
- The inner command to be executed - symmetrically encrypted with the symmetric key.

- A hash of the message - asymmetrically encrypted with the private key of the sender, so the receiver can decrypt it with the public key of the sender upon arrival.

When a CryptoCommand is received, the command checks if the inner command's sender matches the sender of the CryptoCommand itself. It then confirms that the decrypted hash matches the hash it computes locally, and if everything matches up, the command is executed, otherwise the system is notified and shuts down.

8.3 Detection and recovery

Detecting potential intrusion is most likely to happen when receiving a command transmitted over the network. The Communicator only allows CryptoCommands, IsAliveCommand and PublicKeyExchangeCommands to be received, reducing the amount of potential attacks. Upon receiving something else, the system is shut down. IsAliveCommand does not contain any code or data to be executed and can not be exploited. PublicKeyExchangeCommand shuts down the system if the station has already exchanged keys once, and as key-exchange requires human interaction, detecting misuse should be easy. CryptoCommands shuts down the system if the sender is unknown or if the sender hash is invalid.

Another problem that can be detected, is when failing to send a command to a recipient. This is handled differently based on some criteria:

- When the manager fails to send a command to a station, the manager announces to the remaining peers that the station should be removed from their peer-lists.
- When a station fails to send a command to the manager, the station announces to the other stations that they should elect a new manager, and then re-sends the command to the newly elected manager.
- When a station fails to send a command to another station (only likely when it is announcing to other stations that they should elect a new manager), it simply removes the peer from its peer-list.

8.3.1 Electing a new manager

If the manager machine crashes during the election, the system is able to recover by electing another station to be the new manager. Since a crash can potentially happen at any time, there are some required properties the manager election algorithm must have:

- It must be able to elect a unique leader that every station agrees on.
- It must be able to elect the same leader if several elections are initiated, provided the same machines are part of the initiated elections.
- It must terminate.
- It must be relatively fast so it does not impact the users.

To satisfy these requirements we have implemented an algorithm where the station with the highest identifier (e.g. IP address) is elected as manager. If the station with the highest identifier is unreachable, the station with the second highest identifier is elected and so on. This fulfills all

the required properties of our manager election algorithm and gives us a worst case and average case complexity of $O(n)$. This solution requires that each station has a list of all the other stations and their identifiers, that the identifiers do not change during the election and that the identifiers are consistent.

Algorithm 3 Elect a new manager

Require: $!IsActive(CurrentManager)$ {check if the manager is reachable}
1: $L \leftarrow []$
2: $L.Add(IP)$ {add the IP address of this machine since it is not a part of the Peer list}
3: **for all** Peers **do**
4: **if** $IsActive(Peer)$ **then**
5: $L.Add(Peer)$
6: **end if**
7: **end for**
8: Sort L by IP Address
9: **return** $L.First$ {the highest IP address would be the first element in L}

When designing this we considered two alternatives:

Franklin election algorithm - Average complexity $O(n \cdot log(n))$, worst case complexity $O(n^2)$ - This algorithm is a ring election algorithm where each node sends its identity to its two adjacent neighbors, compares its identity with the nearest active neighbors identities and if its identity is not the largest, the node becomes passive. It repeats this until the node with the largest identity receives its own message [3].

Hirschberg-Sinclair algorithm - Average complexity $O(n \cdot log(n))$, worst case complexity $O(n \cdot log(n))$ - This algorithm is also a ring election algorithm and works much like the Franklin algorithm. It operates in waves where each node tries to become the leader by sending a wave k out, if it is the leader when the wave returns it proceeds to the next wave *k+1*. This is repeated until only one node is left which is then elected the leader [4].

The best case scenario for our algorithm ($O(1)$) occurs if the only machine that crashes is the manager machine. If we were to use the Franklin algorithm, this would occur if the station starting the election happens to have the highest identifier. If we used the Hirschberg-Sinclair algorithm the best case scenario would be for every node to have their tokens discarded in the first wave except the node with the highest identifier (this would happen in an ordered ring) but it would still have a $O(n)$ complexity.

We assume that it is unlikely for multiple machines to fail at once, and thus the election of a new manager should run in constant time using our algorithm. If every machine in the network should crash it is more likely that we face an attack than a common error. If we consider the Franklin algorithm the chance of choosing the right starting node is too low and for the Hirschberg-Sinclair algorithm the complexity is too high.

We assume that each election venue has at most 25 machines and that the election of a new manager is not something that happens frequently. Considering that there is a relatively small amount of machines, the choice of election algorithm is not very important, since the speed of the algorithm is unlikely to be noticeable.

8.3.2 Fatal errors

If the system should experience an attack or a major hardware error during the election, the need to switch to a paper based approach arises. Dependant on the situation, different options present themselves. If several computers break down and the amount of operational computers left is not enough, an option would be to print the data as it is, at the time of the breakdown and continue the election by marking the voters manually. With this system there is a slight problem, because the data sets are encrypted during the entire election and the decryption key is held by an entity that is not present in the election venue. While it would be possible to transport the entity to the election venue to decrypt the data set, it could be very time consuming. Another option, that lends itself to this system in a better way, would be to export the already collected data to a portable medium and continue by marking the remaining voters manually. This approach presents the problem of merging the exported data with the manually collected data after the election, which can be prone to errors and can be time consuming.

If the system is the victim of an attack the two solutions above are not sufficient since the printed or exported data set might be compromised. Essentially the gathered data can not be trusted and must be disregarded. While it is still possible to switch to marking the voters manually the digitally gathered data is lost and can not be merged with the manual markings later. The only viable approach would be to have the voters vote again.

8.3.3 Inconsistent data

While this system does everything it can to make an election run as smoothly as possible, we must not overlook a scenario where the data sets on the stations and the manager is inconsistent after the election has ended. The system can not provide any guarantees that this was caused by a software error, a hardware malfunction or a malicious attack. With the current paper based model there are often a few votes unaccounted for compared to the number of people they have marked as having received a ballot and they are ignored i.e. counted as blank votes. There are several solutions to this, each with its own drawbacks and advantages. First, one could ignore the inconsistency and just acknowledge a single data set as being the correct one. This is simple and fast, but gives no guarantee that the data set is correct. Second one could compare the data sets from all the machines and let the majority of identical data sets be considered correct. This is a bit more time consuming, but the guarantee that over half of the machines would have to be compromised to tamper with the data set, is given. Third, the option to do a re-election is present. If one were to identify the flaw in the system, fix it and redo the election all over, a more satisfying result would be achieved. This is both expensive and time consuming, but would be an ideal solution if a correct data set is a requirement.

Aegis DVL does not check the data set for inconsistencies since it should never be able to occur. If a machine tries to change the ballot status of a voter, all the other machines will be updated as well. One thing to take notice of, is that if a station is removed by the manager it should be apparent to the user that the machine will not have a consistent data set anymore since it does not receive updates from the manager anymore. This means that the user interface should have a strong way to inform the user of whether a machine is connected to the manager or not.

8.4 Logging

Logging is a tool to make sure that the execution of the program is easy to inspect. This makes it possible to find out what happened after an election, whether it was a success or something went wrong. We have chosen to store the logs on all the machines locally. They are stored in a database file encrypted with the master password ensuring that it can be accessed at any time. The log file is located in the application directory.

In our implementation we have chosen to have an interface (ILogger) which makes it easy to switch the logging mechanisms if it should be necessary. We have implemented a simple class that inherits from ILogger and can store log entries instead of a framework which would over-complicate this simple operation. For a comparison of some of the most popular logging frameworks see Comparison of .NET Logging Frameworks and Libraries [27].

We have chosen five different logging levels that each indicate a different kind urgency:

Debug - Contextual information used for diagnosis.

Info - Contextual information used to help trace execution.

Warn - Indicates a potential problem in the system.

Error - Indicates a serious problem in the system.

Fatal - Indicates a non-recoverable fatal problem in the system.

We approached our logging with a ”the more the better” mindset and chose to log the following things:

- Every time a ballot status is changed in the database.
- Every time a command is received or sent over the network.
- The start and end of the election.
- Every time the manager announces an event.

When a ballot status is changed, the CPR- and voter number of the changed voter is logged as well. This could be a potential risk, but we make sure that the log is encrypted with the master password and can not be accessed without it.

By logging as much as possible and using the different levels of urgency we create a log which can be filtered to display the information needed by any user. We chose to log as much as possible to prevent future developers from being forced to add more logging-statements to the back-end themselves.

Chapter 9

Comparison with KMD's DVL and other related work

To compare our system to the system developed by KMD, we have listed some of the similarities and differences between the two systems. The comparison is based on the KMD manuals [36][37][38] since we did not get first hand experience with the system.

Similarities:

Both systems operate in a closed network during the election
Both systems require that there is no access to the internet during the election. The system developed by KMD does however use the internet when importing the data, but during the election the connection is severed.

Both systems save their data in simple files
By using the SQLite DBMS only a single database file is used for the data. This is an idea that KMD had as well and it reduces the complexity of the overall structure.

Differences:

The system developed by KMD stores the data in partitions on each machine with a single other machine as backup
While storing the data in partitions is not a problem in itself, the fact that an adversary would only have to attack two machines to gain control over or destroy an entire partition of the voter data is quite the risk. We have chosen to store the data on all the machines thereby minimizing the data loss during a crash.

The system developed by KMD require the machines involved to have static IP addresses
KMD's system requires that each machine has a specified IP address. Our system does not require static IP addresses, but the DiscoverNetworkMachines method only searches in a specified IP range. This is a more flexible solution since no IP configuration is needed.

The system developed by KMD supports letter votes
Our system does not support letter votes, but KMD has gone the extra mile and support letter votes with a separate application. This enables them to process these votes before the actual election and still merge the letter votes with the data at the election venue. The exported data at the end of the election therefore contains all the votes which is desirable.

The system developed by KMD has the option to print replacement voter cards
While it is nice for the voters to have something tangible when they vote, we do not see the use of being able to print additional voter cards. If a voter arrives without his voter card he should be able to identify himself and then be able to vote once his identity has been confirmed. There is no need for him to receive a voter card just to use it quickly thereafter.

The system developed by KMD requires each machine to disable its firewall, screensaver, antivirus and hibernation mode. It also requires that the screen resolution is 1024x768 and that the PC name is static
Our system does not require any of these things, which seem unnecessary and very impractical for the person assigned to set up the system. Disabling the firewall and antivirus will actually lower the security in the event that an unknown attacker enters the network.

The system developed by KMD is designed to be set up the day before the election
This seems like a great idea from a practical standpoint. The person assigned with the set up can do so undisturbed and test the system in advance. The downside is the potential that someone can tamper with the system overnight. The KMD manuals [36][37][38] does not specify anything about the election venue and it would be possible to enter the venue unnoticed and tamper with the machines before the election started. We assessed that the security risk overruled the practical convenience and chose to have the set up on the day of election.

The system developed by KMD is split into two different applications. One for importing data and configuring the system and one for the election itself
This seems like an unnecessary separation of two tasks that are quite closely coupled. It does makes some sense in KMD's system because they wanted to have the system set up a day in advance. If the person assigned to the set up process could import the data and configure the machines ahead of time he might be able to avoid some problems.

The system developed by KMD requires that the configuration files are moved by USB device
The configuration files generated by the importing application must be moved to the manager machine of the election application and put in a specific folder. This seems unnecessary error prone and cumbersome and could easily be solved with an importer in the user interface.

The system developed by KMD uses the Internet to import data
The system uses a technology called CAP-IP to download the data to the machines. While we do not doubt their intentions we wanted to reduce the attack surface as much as possible in our system so we have chosen the data to be transported to the election venue via a portable medium.

The system developed by KMD allows machines to continue the election autonomously if the network should malfunction
While this solution gives a great degree of convenience it decreases the security of the system greatly. If a machine is not connected to the network there is no control with the data set on that machine. An attack would only have to compromise that single machine to produce an inconsistent data set after the election has ended. In our opinion KMD would have been better off if they had chosen a solution where the machine that loses the connection to the network should be excluded from the election.

The system developed by KMD has two different levels of ambition for handling errors
This is an interesting notion and shows that KMD has a realistic view of how election venues differ from each other. Ideally every venue would adhere to the high level of ambition, but in reality this is not possible. Our system does not have such a notion, but it would be a consideration for further development.

The system developed by KMD only requires that the election secretary logs into the system before the ballot statuses of the voters can be changed
This presents a potential security risk. If we assume that the election secretary logs into the system at the start of the election and then later needs to get a cup of coffee, nothing is stopping anyone from editing the statuses of the voters during that time. We have chosen to have the election secretary type the master password each time the ballot status of the voter needs to be changed. While this might be considered an inconvenience, it increases the security.

Our system does not have an end-to-end voter auditable trail [39] which allows for voters to verify that their voter has been counted correctly or in our case that the voter has been marked as having received a ballot. Systems like Punchscan [40] and Scantegrity [41] implement this and this should be considered for further development of our system although focus on a trail for the votes is more interesting than the voter cards. One can argue that if there is a trail to the vote, a trail to the voter card is redundant.

Another consideration is whether or not to have actual voting machines dedicated to only the task at hand. Voting machines are available from vendors such as Dominion Voting [42], but can be expensive compared to a normal PC. The advantages of using a voting machine is that it is harder to compromise since the user interface and functionality is smaller than that of a PC. The disadvantages is the price and the fact that updates to these machines comes from a single commercial vendor who might not provide transparency for their system. This could make it hard to verify whether or not the system works as intended for anyone outside the vendor company.

Compared to the system developed by KMD, our system has less restrictions and a more robust way of storing the voter data. While the KMD system might have some practical aspects our system lacks, the robustness and security of our system is superior.

Chapter 10

User Manual and Users

Contrary to KMD's user manuals [36][37][38] we have not split our user manual into sections based on the roles of the people handling the system, but instead based on the different parts of the system. This is because we believe that any single person can potentially handle the entire system from setup to completion of the election. In reality this is limited by the election secretary which is the only person who should hold the master password needed for some of the larger decisions in the election.

To run the program one must have the appropriate DBMS installed. In our case this means that the ADO.NET 2.0 Provider for SQLite (link found in appendix 17.3 User manual) must be installed prior to the running of the application. As a second requirement a PDF reader must be installed if the user manual, found in the "Bruger manual" item under the "Hjælp" menu, is to be displayed. This is optional although the user should be aware that the user manuals can not be viewed without it.

In our current solution we want the election secretary to be the only individual who knows the master password to maximize the security. By only having one individual that know it, we do not need to trust the entire election staff, but only a single person. However if the master password was to be shared between several individuals one should be aware that entries in the log that could only have been done by an individual possessing the master password can reflect different persons. This is not something we can easily enforce in the system and we trust that the election secretary is trustworthy.

Since the master password is needed to mark a voter by CPR number only, which should only happen when a voter has lost or forgotten his voter card, we realized that if a large number of these voters appeared at the same time this might create a bottleneck since only a single person can mark these voters. After further investigation we discovered that this has not previously been a problem in Denmark, as few voters forget or lose their voter cards. If this were to become a problem, one could add another tier of election staff between the election official and election secretary. This new tier would have a separate password for each member and would be able to have all the rights of the election official with the added benefit of being able to mark voters by CPR number only.

The user interface in our application is supposed to be for demonstration purposes only. We wanted to focus on making a system with an easily replaceable user interface. This does not

mean that the user interface is not functional, but the aesthetics of it can be improved.

Chapter 11

Testing

Testing the software gives us some confidence that it works correctly. Having the tests cover 100% of the code-base while asserting that it functions as intended, gives us full confidence that the code does not *always* fail. The more thorough tests, the higher confidence that the software works as expected. We also verified our synchronization algorithm using UPPAAL, see appendix 17.4 UPPAAL.

The scanner and voter card generator was tested during the development but these tests remain undocumented. Since there is no code for the scanner and we did not write any of the code for the voter card generator, we found it unnecessary to tests these features in a systematic manner.

11.0.1 Test strategy

As a primary means of testing we have created unit tests using the NUnit testing framework [30]. For tracking code-coverage, we have used JetBrains dotCover [31]. We initially set requirements for the coverage of our tests, by dividing the tests into domains and setting coverage requirements. Ideally we would like 100% coverage, but in some cases it is impractical, so we settled for 90% coverage on most of the domains. The tests should also be thorough, but it is hard to specify this in requirements. Due to time constraints, some of the tests are not as thorough as we would have liked.

We would also have liked to have run PEX [32] on our system. We tried running PEX briefly, but it generated a lot of tests that failed, and we did not have time to identify which tests were problems that needed fixing, and which were PEX being unable to generate good tests. Ideally, all of PEX' failed tests should be corrected, or at least analyzed, but as we had good test coverage from our hand-written tests, we did not include the PEX tests.

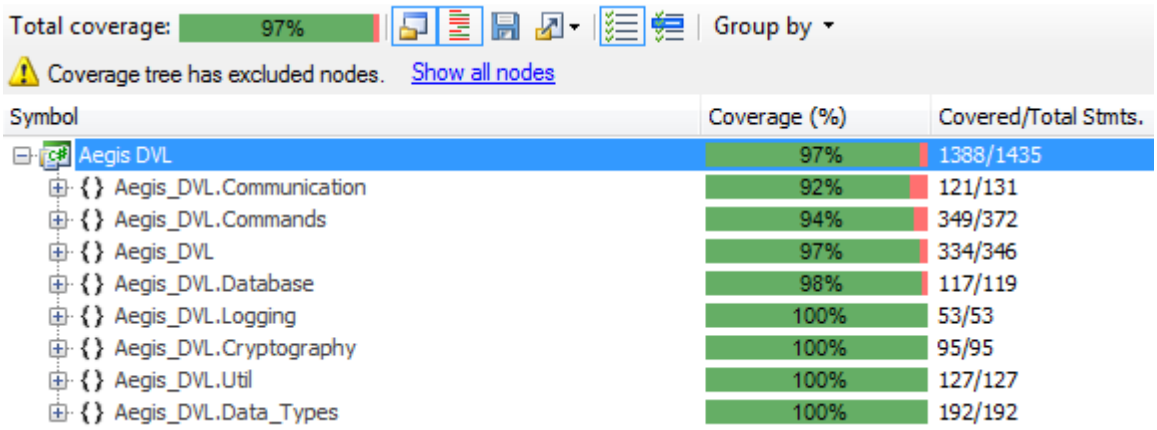
The unit tests were only written for the Aegis DVL system and not the user interface. The user interface was black-box tested. We consider white-box (unit testing) testing to be a more reliable way of testing, but also more time-consuming. We could have unit tested some of the user interface, but other parts of it would be problematic. Ultimately, as the user interface is only meant for demonstration purposes, we decided only to black-box test it.

11.0.2 Results

Test results

The coverage results exclude parts of the system. It excludes some of the generated Entity Framework code, as we have not written it nor used it beyond what was covered. We've also excluded some Finalize methods that were not being run due to IDisposable being implemented. The Finalize methods were not written by us, either. Some of the code was wrongfully marked as not being covered due to reasons unknown. This mostly covered lambda expressions in code contracts.

Coverage results



59/72 of the user interface blackbox tests passed. To view the tests in detail, see appendix 17.1 User interface tests. Most of these bugs are insignificant and can be easily repaired. They do not interrupt the normal workflow but are more of an inconvenience to the users. However, they should still be fixed before making the application publicly available since some of the bugs will crash the program completely.

11.0.3 Known bugs

Our testing revealed some bugs listed here:

Known bugs

Bug	**Severity**
A station will never know it has been removed from the group, only the manager and all other stations will.	Major
When you add a station in the ManagerOverviewPage, it gets connected, but the election never starts as it is busy receiving the SyncCommand.	Major
"Random" IOExceptions : Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host.	Major
You can promote a machine you are not connected to in the ManagerOverviewPage which results in the manager being lost.	Major
"Start valg" works, but the listener on the stations should not be busy executing other commands (like the SyncCommand after a public key-exchange), as it will not receive other commands during execution.	Major
If a station types in the proper password during a public key-exchange, but the manager cancels, then the station will have the manager's address and public key, but not the other way around. This will make following public key-exchange requests fail unless you re-create the station.	Minor
ElectNewManagerCommand should never be send to the manager.	Minor
You can only paste in 9 chars, and not the 10 of a CPR number in the UI.	Minor
We have on rare occasions experienced this exception on the manager machine: *The CLR has been unable to transition from COM context 0x1b7ae0f0 to COM context 0x1b7ae340 for 60 seconds. The thread that owns the destination context/apartment is most likely either doing a non pumping wait or processing a very long running operation without pumping Windows messages. This situation generally has a negative performance impact and may even lead to the application becoming non responsive or memory usage accumulating continually over time. To avoid this problem, all single threaded apartment (STA) threads should use pumping wait primitives (such as CoWaitForMultipleHandles) and routinely pump messages during long running operations.*. It seems to mainly have been thread-deadlocking that has caused it, and we have not been able to consistently recreate it.	Minor
If you click "Opdater" in the user interface while it is already updating, you will get an ObjectDisposedException. This is because the DiscoverNetworkMachines method uses the threadpool.	Minor
CPR numbers written in the user interface should be within the uint32 limits. Ideally, we should add more checks to the user interface, like making sure that the first two digits do not exceed 31, the next two digits do not exceed 12 and so on.	Minor
If multiple machines try to request the same ballot at the same time, only one is handed out, but no error message is shown on the other machines.	Minor
If you click "Marker vælger" or "Afslut", any entered master password is considered wrong if you are in a window before the BallotRequestPage on a station or before the OverviewPage on the manager.	Minor
If you select an invalid key during load, an exception is thrown in the DataLoadPage.	Minor
If you try to add a station you are already connected to, an exception is thrown in the OverviewPage and ManagerOverviewPage.	Minor

When you have removed a station, the user interface list is not updated before you click ”Opdater”.	Minor

While most of the known bugs are minor and easily repairable we identified five major bugs. These bugs interrupts the normal workflow when using the application and must be fixed before using the application in a real world environment

Chapter 12

Future Development

When we started this project we were aware that gaining access to the government databases in Denmark was something that we did not want to pursue. We aimed to develop a system where another developer could easily adapt it to fit new database structures and communication method. To promote modularity and make it easy to exchange one part of the system without affecting other parts we made the following interfaces:

- ICommunicator
- ICommand
- ICrypto
- IDatabase
- IScanner
- IDvlUi
- ILogger

We also wanted to make a logging system where we logged as much information as possible. It could seem to be hard to find the information you are searching for, but with modern log analysis tools this can be achieved without too much of a hassle. We would rather log too much information and have future developers filter it, than log too little and force them to insert their own log statements all over the code.

12.1 Improvements

As a starting point for future development we have made a list of improvements would like to have done ourselves were we given more time:

- System
  - For the system to be able to support letter votes prior to the election. This might benefit from having its own project and application but many of the principles discussed in this paper could be relevant.
  - Construct an easy way for users to access the log and filter it.

  - Be able to adjust the IP range and timeout for the DiscoverNetworkMachines method in Station from the user interface.
  - Make an installer that installs SQLite and a PDF reader, such as Adobe acrobat reader, along with the application.
  - Modify the logging system to implement distributed logs instead of locally stored logs.
  - Modify the application in such a way that it would run as a service and require administrator rights to close.
  - Create a possibility to test the system before the election starts. Potentially done via a test voter.
  - Implement a message queue system in the manager communication layer.

- User Interface
  - Make sure that scanned voter number will be entered in the right text box regardless of focus.
  - For the user interface to be able to populate the lists of station in the OverviewPage and ManagerOverviewPage automatically and update it every ten seconds.
  - Remove the "Opdater" buttons on the OverviewPage and ManagerOverviewPage.
  - Make the "Tilføj", "Fjern" and "Gør til Manager" buttons in the OverviewPage and ManagerOverviewPage inactive when nothing is selected, instead of the current solution when nothing happens when they are pressed.
  - Bind the "Enter" key to the correct button in the ManagerOverviewPage dependant on which text boxes were filled.
  - For the user interface to be able to mark the correct station as not connected when the "Fjern" button is pressed in the OverviewPage and ManagerOverviewPage instead of populating the entire list again.
  - Construct a user interface for generating voter cards.
  - Make the AcceptManagerDialog, AcceptStationDialog and CheckMasterPasswordDialog focus the text box.

Chapter 13

Glossary

Election venue One of the venues where the election is held. Each venue has it own set of machines and election personnel.

Station A machine where voters can scan or type in their voter numbers and CPR numbers and are handed a ballot if they are eligible.

Manager A machine that manages the stations in the network. The manager machine can add or remove stations from the network during the election. The election data is imported and exported from the manager machine. The manager machine is also responsible for starting and ending the election at the appropriate times.

Voter A person eligible for voting.

Voter card Each voter receives a voter card prior to the election. The voter card contain the voter number, name and election venue of the voter and is used to verify whether the voter is eligible to vote at a specific venue. When the voter wants to vote he has to present the voter card to receive a ballot.

Voter number A unique number identifying a specific voter during an election.

Ballot When a voter has been verified as eligible to vote he receives a ballot used to cast a vote.

Election official A normal poll worker that does not know the master password. The job of the election official is to hand out ballots to the eligible voters when the system has confirmed that it is OK.

Election secretary The person responsible for a single election venue. Each election venue has one election secretary that holds the master password for that venue.

Master password A password generated before the election starts and held by the election secretary. It is used to start an election, end an election, register a voter only with his CPR number and access the log database.

Chapter 14

Reflection

When designing a software solution that focuses on security one must be aware that no system is 100% secure. Every time a new layer of security is added the responsibility is moved from one entity to another, whether this is a part of the system or an actual person (or multiple persons). It all comes down to which entities you trust. In this system we assume that the election secretary and the entity responsible for partitioning and collecting the data are both trustworthy sources. If any of these were to have malicious intent, they could easily jeopardize the election. This could be solved by adding a new layer of security and having a new entity control the privileges of the election secretary and the partitioning and gathering entity. This poses the problem of whether we trust the new controlling entity, and illustrates that adding additional layers of security is not always beneficial.

A desirable way to deal with this is distributed security. If several entities with different stakes control the security together it becomes more robust. As an example, a married couple might share a bank account. The husband does not trust the wife not to spend all the money on shoes, and the wife does not trust the husband not to spend it all on wine, but they need to be able to extract money from the bank account for shared needs. If they both have a part of the account password, they can only extract money from the account when both of them are present. This prevents each of them from emptying the bank account on their own. The same principle could be applied to the election venue, with members from opposing political parties, both not wanting the other to inappropriately manipulate the election.

When implementing the security in our system, we realized just how hard it actually is to implement, and how easy it is to implement it wrong. We initially considered using SSL and PGP/GPG with OpenSSL [60] as using verified security approaches gives a greater sense of trust, but the documentation for OpenSSL.NET [61] was severely lacking. We eventually switched to using Bouncy Castle, where the documentation was better, but not great. Its greatest strength was probably the fact that it was a .NET implementation, and not merely a C wrapper, like OpenSSL.NET. In the end, we decided to implement our own secure communication. This was partially done due to not requiring all of the functionality of SSL or PGP/GPG, but also because of Bouncy Castle lacking some functionality, such as a SSL server, and the fact that the PGP/GPG implementation was clunky. Using a lot of the concepts of PGP/GPG, we do believe our secure communication is actually secure.

Chapter 15

Conclusion

We believe the project has been a success. We successfully built a distributed digital voter list system with no single point of failure, that uses secure network communication and make use of encryption to secure personal sensitive data. The system was fully documented using the BON specification language, and was created using design by contract. A part of the system was also verified using the model checker UPPAAL. The system was also tested thoroughly, with a total of 97% code coverage. Though there are problems with the system that need to be fixed if it were to be used in a real election, the theory and design decisions are sensible and there is a solid foundation that can be developed from. With further development, we definitely believe the system could replace the system made by KMD. The primary requirements were fulfilled, and some of the secondary as well.

Primary requirements:

Features
All of the requirements in this category were met. We have constructed a system with a graphical user interface where at least one manager machine and three station machines must be present.

Code requirements
All of the testing and code requirements were met. The system is documented and tested using unit tests, black box tests and code contracts.

The system
All of the system requirements were met. The system is able to scan and print voter cards, it allows the extraction of the full data set at any given time during the execution of the application, and it allows voters to use any of the machines in the election venue.

Secondary goals (optional):

It should be faster to use the system that using the current paper-based model.
We did not test the speed of our system compared to the current paper based system, but this could be an important metric when an optimal user interface is constructed. We advise that speed should be a part of the user test conducted when testing a new user interface.

The system should be able to generate a list of all the voters of the election place and whether they have voted or not and print it.
This requirement was not met, and in retrospect it should not have been a goal. Our

system has had a strong focus on security, and all the voter data is encrypted. Being able to print all the voter data could be considered a security flaw, and private sensitive data such as CPR numbers could needlessly be exposed. Nevertheless, the PDF generator code is able to generate a list of voter names and voter numbers, but this feature is never used.

The graphical user interface should be easy to learn and use.
We did not test the usability of the user interface since it is only meant for demonstration purposes. If a new user interface is created, there should be a focus on the ease of learning and ease of use.

The system should support letter votes.
This requirement was not met, but the possibility for gathering the letter votes beforehand and passing the voter data to our system is present, thereby eliminating the need to merge the data later on. However, this would require that the letter votes were partitioned in the same way as the voter data for each election venue.

Use a data flow analysis tool to reason about correctness of the data flow in the system.
We used the model checking tool UPPAAL [28] to reason about the synchronization algorithm in the system. UPPAAL could also be used to reason about additional parts of the system to ensure its correctness.

Use an analysis tool to reason about the cryptographic protocol used.
This requirement was not met, but would be a great addition to the security guarantee the system provides.

Chapter 16

References

[1] Applied information security: A hands-on approach - David Basin, Patrick Schaller, Michael Schläpfer - Springer-Verlag Berlin Heidelberg 2001

[2] Distributed Algorithms - Nancy A. Lynch - Morgan Kaufmann Publishers Inc. 1996

[3] Leader Election Algorithm in Anonymous Rings: Franklin Goes Probabilistic - Rena Bakhshi - Milan, September 9, 2008 - retrieved from `http://www.few.vu.nl/~rbakhshi/papers/TCS08talk.pdf` on 12th March 2012

[4] Leader Election in rings - Marco Aiello, Eirini Kaldeli - University of Groningen 2009 - retrieved from `http://www.cs.rug.nl/~eirini/DS_slides/leader_election.pdf` on 12th March 2012

[5] Attack Modeling for Information Security and Survivability - Andrew P. Moore, Robert J. Ellison, Richard C. Linger - March 2001 - retrieved from `http://www.cert.org/archive/pdf/01tn001.pdf` on 12th March 2012

[6] Attack Trees: Modeling security threats - Bruce Schneier - Dr. Dobb's Journal December 1999 - retrieved from `http://www.schneier.com/paper-attacktrees-ddj-ft.html` on 12th March 2012

[7] Creating Secure Systems through Attack Tree Modeling 10 June 2003 - retrieved from `http://www.amenaza.com/downloads/docs/5StepAttackTree_WP.pdf` on 12th March 2012

[8] Improving Web Application Security: Threats and Countermeasures - J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan - Microsoft Corporation June 2003 - retrieved from `http://msdn.microsoft.com/en-us/library/ff648644.aspx` on 12th March 2012

[9] Database Encryption: An Overview of Contemporary Challenges and Design Considerations - Erez Shmueli, Ronen Vaisenberg, Yuval Elovici, Chanan Glezer - SIGMOD Record, September 2009 - retrieved from `http://www.ics.uci.edu/~ronen/Site/Research_files/p29.surveys.shmueli.pdf` on 12th March 2012

[10] 24 DBMS_CRYPTO - Oracle Database PL/SQL Packages and Types Reference 10g Release 2 (10.2) Part Number B14258-02 - retrieved from `http://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_crypto.htm` on 12th March 2012

[11] Database Encryption in SQL Server 2008 Enterprise Edition - Sung Hsueh - Microsoft, February 2008 - retrieved from http://msdn.microsoft.com/en-us/library/cc278098(v=sql.100).aspx on 12th March 2012

[12] Protect Sensitive Data Using Encryption in SQL Server 2005 - Don Kiely - Microsoft, December 2006 - retrieved from download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/SQLEncryption.doc on 12th March 2012

[13] 11.13. Encryption and Compression Functions - retrieved from http://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html on 12th March 2012

[14] Encrypting an Access Database - Mike Chapple - retrieved from http://databases.about.com/od/productinfo/a/encryption.htm on 12th March 2012

[15] PostgreSQL 8.1.23 Documentation : 16.6. Encryption Options - retrieved from http://www.postgresql.org/docs/8.1/static/encryption-options.html on 12th March 2012

[16] The SQLite Encryption Extension (SEE) - retrieved from http://www.hwaci.com/sw/sqlite/see.html on 12th March 2012

[17] SQLite Home Page - retrieved from http://www.sqlite.org on 18th March 2012

[18] How to protect data in Firebird database? - retrieved from http://www.firebirdfaq.org/faq160/ on 12th March 2012

[19] Adaptive Server Enterprise 15.0 ¿ ASE 15.0 with Encrypted Columns - retrieved from http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc00412_1500/html/Encrypt_Guide/BAJCAIHA.htm on 12th March 2012

[20] Encrypting Data Values in DB2 Universal Database - Bruce Benfield, Richard Swagerman - International Business Machines Corporation, 2001 - retrieved from http://www.ibm.com/developerworks/data/library/techarticle/benfield/0108benfield.html on 12th March 2012

[21] MongoDB - retrieved from http://www.mongoDB.org on 19th March 2012

[22] The Apache CouchDB Project - retrieved from http://couchdb.apache.org/ on 19th March 2012

[23] Redis - retrieved from http://redis.io on 19th March 2012

[24] Distributed snapshots: determining global states of distributed systems - K. Mani Chandy & Leslie Lamport - ACM Transactions on Computer Systems, Vol. 3, No. 1, February 1965. - retrieved from http://research.microsoft.com/en-us/um/people/lamport/pubs/chandy.pdf on 10th April 2012

[25] Why Multicast Protocols (Don't) Scale : An Analysis of Multipoint Algorithms for Scalable Group Communication - Eve M. Schooler - California Institute of Technology, 2001 - retrieved from http://thesis.library.caltech.edu/3236/11/thesis.pdf on 10th April 2012

[26] SyncAlgorithm - retrieved from http://code.google.com/p/nsync/wiki/SyncAlgorithm on 10th April 2012

[27] Comparison of .NET Logging Frameworks and Libraries - retrieved from http://www.dotnetlogging.com/comparison/ on 16th April 2012

[28] UPPAAL home - retrieved from http://www.uppaal.org/ on 7th May 2012

[29] Digital Voter Registration System - Christian Olsson, Kåre Sylow Pedersen and Henrik Haugbølle - IT University of Copenhagen, 14th December 2011

[30] NUnit Home - retrieved from http://www.nunit.org/ on 8th May 2012

[31] Code coverage tool for .NET :: dotCover http://www.jetbrains.com/dotcover/ on 11th May 2012

[32] Pex, Automated White box Testing for .NET - retrieved from http://research.microsoft.com/en-us/projects/pex/default.aspx on 18th May 2012

[33] Business Object Notation (BON) - Kim Waldn, Enea Data - Chapter 10 in "Handbook of Object Technology", CRC Press 1998 - retrieved from http://www.bon-method.com/handbook_bon.pdf on 10th May 2012

[34] Code Contracts - retrieved from http://msdn.microsoft.com/en-us/library/dd264808.aspx on 10th May 2012

[35] Applying "Design By Contract" - Bertrand Meyer - October 1992 - retrieved from http://se.ethz.ch/~meyer/publications/computer/contract.pdf on 10th May 2012

[36] Systembeskrivelse KMD Digital Valgliste Version 2.1.0 - KMD A/S 05-09-2011 - retrieved from http://nykundenet.kmd.dk/systembrugere/valg/Valgudskrivning/Vejledninger/Digital%20Valgliste.%20Systembeskrivelse.%20Version%202.1.0.pdf on 10th May 2012

[37] Kom godt i gang KMD Digital Valgliste. Tekniker Version 2.1.0 - KMD A/S 05-09-2011 - retrieved from http://nykundenet.kmd.dk/systembrugere/valg/Valgudskrivning/Vejledninger/Kom%20godt%20i%20gang.%20Digital%20Valgliste.%20Tekniker.%20Version%202.1.0.pdf on 10th May 2012

[38] Installationsvejledning til KMD Digital Valgliste Konfiguration Version 2.2 - KMD A/S - retrieved from http://nykundenet.kmd.dk/systembrugere/valg/Valgudskrivning/Vejledninger/Installationsvejledning%20til%20KMD%20Digital%20Valgliste%20Konfiguration%20Version%202.2.pdf on 10th May 2012

[39] E-Voting Technology Glossary - retrieved from http://whatis.techtarget.com/glossary/e-voting-glossary.html on 11th May 2012

[40] Punchscan see your vote count - retrieved from http://www.punchscan.org/ on 11th May 2012

[41] Scantegrity - retrieved from http://www.scantegrity.org/ on 11th May 2012

[42] Dominion Voting is a different kind of election partner - retrieved from http://www.dominionvoting.com/ on 11th May 2012

[43] Mediator Design Pattern in C# and VB.NET - retrieved from http://www.dofactory.com/Patterns/PatternMediator.aspx on 15th May 2012

[44] The International PGP Home Page - retrieved from http://www.pgpi.org/ on 15th May 2012

[45] Command Design Pattern in C# and VB.NET - retrieved from http://www.dofactory.com/Patterns/PatternCommand.aspx on 15th May 2012

[46] Four-eye principle / Planning and organization - retrieved from http://www.economypoint.org/f/four-eye-principle.html on 15th May 2012

[47] ADO.NET 2.0 Provider for SQLite - retrieved from http://sourceforge.net/projects/sqlite-dotnet2/ on 18th May 2012

[48] What is object-relational mapping (ORM)? - retrieved from http://searchwindevelopment.techtarget.com/definition/object-relational-mapping on 18th May 2012

[49] The GNU Privacy Guard - retrieved from http://www.gnupg.org/ on 18th May 2012

[50] What is SSL? SSL Certificate Basics - retrieved from http://www.sslshopper.com/what-is-ssl.html on 18th May 2012

[51] The Legion of the Bouncy Castle C# Cryptography APIs - retrieved from http://www.bouncycastle.org/csharp/ on 18th May 2012

[52] RSA Algorithm - retrieved from http://www.di-mgt.com.au/rsa_alg.html on 18th May 2012

[53] Optimal Asymmetric Encryption: How to Encrypt with RSA - Mihir Bellare, Phillip Rogaway - Springer-Verlag, 19. nov 1995 - retrieved from http://cseweb.ucsd.edu/users/mihir/papers/oae.pdf on 18th May 2012

[54] AES Explained - retrieved from http://x-n2o.com/aes-explained on 18th May 2012

[55] Secure Programming Cookbook for C and C++, section 5.4.3.2 - Matt Messier, John Viega - O'Reilly - July 2003

[56] PKCS #7: Cryptographic Message Syntax - retrieved from http://tools.ietf.org/html/rfc2315 on 18th May 2012

[57] Secure Programming Cookbook for C and C++, section 5.4 - Matt Messier, John Viega - O'Reilly - July 2003

[58] Recommendation for Key Management Part 1: General (Revised) - Elaine Barker, William Barker, William Burr, William Polk, Miles Smid - NIST Special Publication March 2007 - retrieved from http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf on 21th May 2012

[59] How secure is AES against brute force attacks? - Mohit Arora - retrieved from http://www.eetimes.com/design/embedded-internet-design/4372428/How-secure-is-AES-against-brute-force-attacks- on 21th May 2012

[60] OpenSSL: The Open Source toolkit for SSL/TLS - retrieved from http://www.openssl.org/ on 22th May 2012

[61] OpenSSL.NET - retrieved from http://openssl-net.sourceforge.net/ on 22th May 2012

Chapter 17

Appendix

17.1 User interface tests

UI Tests

No	Task	Expected Behavior	Did it behave as expected	Errors
-	TypeChoicePage	-	-	-
1	Push the Station button on the TypeChoicePage	Redirection to the WaitingForManagerPage	Yes	None
2	Push the Manager button on the TypeChoicePage	Redirection to the MasterPasswordPage	Yes	None
3	Push the Afslut button on the TypeChoicePage	The application closes	Yes	None
-	Menus	-	-	-
4	Choose User manual under the Help menu	the user manual opens as a .pdf file	Yes	None
5	Choose Exit under the File menu	A prompt asks for the master password and the application closes if it correct	No	The master password is always be false if you are in TypeChoicePage, WaitingForManagerPage, MasterPasswordPage and DataLoadPage. This is becsuse the station object have not been initialized.

6	Choose Export Data under the File Menu	two prompt appears, one asking for the master password and one asking for the destination of the data. If both are valid the data is exported to the location.	Yes	None
7	Choose Mark Voter under the File Menu	two prompts appears one allowing you to type the CPR number of a voter and one asking you for the master password. If the master password is correct a prompt shows whether or not the voter is eligible for a ballot	Yes	None
-	DataLoadPage	-	-	-
8	Press Næste on the DataLoadPage with data and key selected in the right format	redirection to the OverviewPage	Yes	None
9	Press Næste on the DataLoadPage with data selected in the right format but the key in the wrong format	A prompt telling you the import was not successful	No	An Exception is thrown
10	Press Næste on the DataLoadPage with both data and key selected in the wrong format	A prompt telling you the import was not successful	No	An Exception is thrown
11	Press Næste on the DataLoadPage with key selected in the right format but the data in the wrong format	A prompt telling you the import was not successful	Yes	None
12	Press Næste on the DataLoadPage with no key and no data selected	A prompt telling you the import was not successful	Yes	None
13	Pressing the Tilbage button on the DataLoadPage	redirection to the TypeChoicePage	Yes	None
-	MasterPasswordPage	-	-	-
14	Entering the MasterPasswordPage	a random generated password is shown	Yes	None

15	Pressing Tilbage on the MasterPasswordPage	redirection to the TypeChoicePage	Yes	None
16	Pressing Næste on the MasterPasswordPage	redirection to the DataLoadPage	Yes	None
-	WaitingForManagerPage	-	-	-
17	While on the WaitingForManagerPage a manager tries to connect	A prompt asking for a password to be typed appears. If this is correct a similar prompt appears on the manager and the password is shown on the station	Yes	None
18	While on the WaitingForManagerPage a manager is connected	The Page displays the text Venter påat valget starter	Yes	None
19	While on the WaitingForManagerPage the election is started	redirection to BallotRequestPage	Yes	None
20	Press Tilbage while on WaitingForManagerPage	redirection to TypeChoicePage	Yes	None
-	BallotRequestPage	-	-	-
21	Press Færdig with a valid voter number and CPR number in the appropriate text boxes	A prompt saying that the voter can be handed a ballot appears	Yes	None
22	Press Færdig with an invalid voter number and CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
23	Press Færdig with a valid voter number and but an invalid CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
24	Press Færdig with an invalid voter number and a valid CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
25	Press Færdig with no voter number and a valid CPR number in the appropriate text boxes	You can not press the Færdig button	Yes	None

26	Press Færdig with a valid voter number and no CPR number in the appropriate text boxes	You can not press the Færdig button	Yes	None
27	Press Færdig with no voter number and no CPR number in the appropriate text boxes	You can not press the Færdig button	Yes	None
28	Press Færdig with a valid voter number and a valid CPR number in the appropriate text boxes, that has already voted	A prompt saying that the voter can not be handed a ballot appears	Yes	None
29	Press Færdig with a valid voter number and a valid CPR number in the appropriate text boxes but not enough stations are connected	You can not press the Færdig button and a label showing that not enough stations are connected appears	Yes	None
-	EndedElectionPage	-	-	-
30	Press the Gennemse button in the EndedElectionPage	a file browser appears and lets you choose a destination, if you do notchoose one nothing appears in the text box	Yes	None
31	Press the Eksporter button with no destination selected in the EndedElectionPage	you can not press the Eksporter button	Yes	None
32	Press the Eksporter button with a destination selected in the EndedElectionPage	The data is exported to the selected destination	Yes	None
-	OverviewPage	-	-	-
33	Press the Opdater button in the OverviewPage	A progress bar appears indicating that the list is updating. When it is done the list is updated	Yes	None
34	Press the Opdater Button in the OverviewPage while it is updating	the old update is canceled and a new update of the list starts	No	a ObjectDisposedException is thrown
35	Press the Tilbage button in the OverviewPage	redirection to the DataLoadPage	Yes	None

36	Press the Tilføj button with nothing selected in the OverviewPage	Nothing happens	Yes	None
37	Press the Fjern button with nothing selected in the OverviewPage	Nothing happens	Yes	None
38	Press the Tilføj button with a station you are already connected to, selected in the OverviewPage	Nothing happens	No	an Exception is thrown
39	Press the Fjern button with a station you are not connected to, selected in the OverviewPage	Nothing happens	Yes	None
40	Press the Tilføj button with a station you are not connected to, selected in the OverviewPage	a password appears on the screen and a prompt to type in this password appears on the station	Yes	None
41	A station replies to your request to add it in the OverviewPage	a prompt appears on your screen and if you type in the correct password the station appears as connected in the list	Yes	None
42	Press the Fjern button with a station you are connected to, selected in the OverviewPage	The station appears as not connected in the list	No	while it is removed, it appears in the list as not connected only after the list has been updated.
43	Press the Start Valg button in the OverviewPage while you are connected to an amount of stations less than the required amount	a box appears telling you that you can not start the election without connecting to more machines	Yes	None
44	Press the Start Valg button in the OverviewPage while you are connected to the required amount of stations or more	redirection to the ManagerOverviewPage. All the connected stations redirected to the BallotRequestPage	Yes	None
-	ManagerOverviewPage	-	-	-

45	Press Færdig with a valid voter number and CPR number in the appropriate text boxes	A prompt saying that the voter can be handed a ballot appears	Yes	None
46	Press Færdig with an invalid voter number and CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
47	Press Færdig with a valid voter number and but an invalid CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
48	Press Færdig with an invalid voter number and a valid CPR number in the appropriate text boxes	A prompt saying that the voter can not be handed a ballot appears	Yes	None
49	Press Færdig with no voter number and a valid CPR number in the appropriate text boxes	You can not press the Færdig button	No	A prompt appears saying that voter can not receive a ballot
50	Press Færdig with a valid voter number and no CPR number in the appropriate text boxes	You can not press the Færdig button	Yes	None
51	Press Færdig with no voter number and no CPR number in the appropriate text boxes	You can not press the Færdig button	Yes	None
52	Press Færdig with a valid voter number and a valid CPR number in the appropriate text boxes, that has already voted	A prompt saying that the voter can not be handed a ballot appears	Yes	None
53	Press Færdig with a valid voter number and a valid CPR number in the appropriate text boxes but not enough stations are connected	You can not press the Færdig button and a label showing that not enough stations are connected appears	Yes	None
54	Press Kun CPR with a valid CPR number in the appropriate text box	A prompt saying that the voter can be handed a ballot appears after you have typed the master password	Yes	None

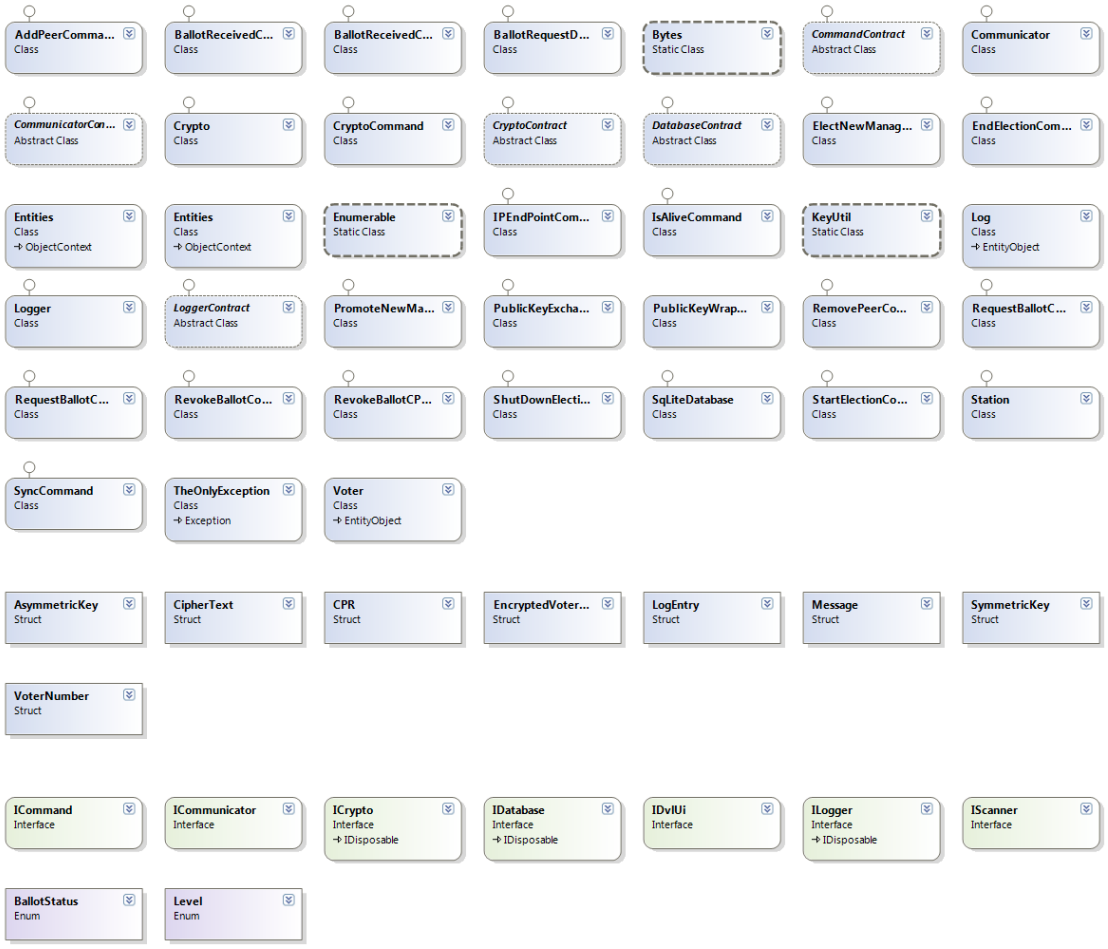
55	Press Kun CPR with an invalid CPR number in the appropriate text box	A prompt saying that the voter can not be handed a ballot appears after you have typed the master password	Yes	None
56	Press Færdig with no CPR number in the appropriate text box	You can not press the Kun CPR button	Yes	None
57	Press Kun CPR with a valid CPR number in the appropriate text boxes but not enough stations are connected	You can not press the Kun CPR button and a label showing that not enough stations are connected appears	Yes	None
58	Press the Opdater button in the ManagerOverviewPage	A progress bar appears indicating that the list is updating. When it is done the list is updated	Yes	None
59	Press the Opdater Button in the ManagerOverviewPage while it is updating	the old update is canceled and a new update of the list starts	No	a ObjectDisposedException is thrown
60	Press the Tilføj button with nothing selected in the ManagerOverviewPage	Nothing happens	Yes	None
61	Press the Fjern button with nothing selected in the ManagerOverviewPage	Nothing happens	Yes	None
62	Press the Tilføj button with a station you are already connected to, selected in the ManagerOverviewPage	Nothing happens	No	an Exception is thrown
63	Press the Fjern button with a station you are not connected to, selected in the ManagerOverviewPage	Nothing happens	Yes	None
64	Press the Tilføj button with a station you are not connected to, selected in the ManagerOverviewPage	a password appears on the screen and a prompt to type in this password appears on the station	Yes	None

65	A station replies to your request to add it in the ManagerOverviewPage	a prompt appears on your screen and if you type in the correct password the station appears as connected in the list. The station is redirected to the BallotRequestPage	No	the station is never redirected to the BallotRequestPage
66	Press the Fjern button with a station you are connected to, selected in the ManagerOverviewPage	The station appears as not connected in the list	No	while it is removed, it appears in the list as not connected only after the list has been updated.
67	Press the Gør til Manager button while nothing is selected in the ManagerOverviewPage	Nothing happens	Yes	None
68	Press the Gør til Manager button while a station you are not connected to, is selected in the ManagerOverviewPage	Nothing happens	No	The station never gets promoted but the manager gets demoted to a station
69	Press the Gør til Manager button while a station you are connected to, is selected in the ManagerOverviewPage	the manager gets demoted to at station and the station becomed the new manager. Redirect to BallotRequestPage for manager and redirect to ManagerOverviewPage for station	Yes	None
70	Press the Afslut Valg button int he ManagerOverviewPage	after having typed the correct master password, redirect to the EndedElectionPage. All stations close their applications	Yes	None
-	Election and crashes	-	-	-
71	During the election, sever the connection to the manager	a new manager is elected and promoted	No	a new manager is elected correctly but not at the time the severing occurs, but on the next action requiring network traffic taken by any station.

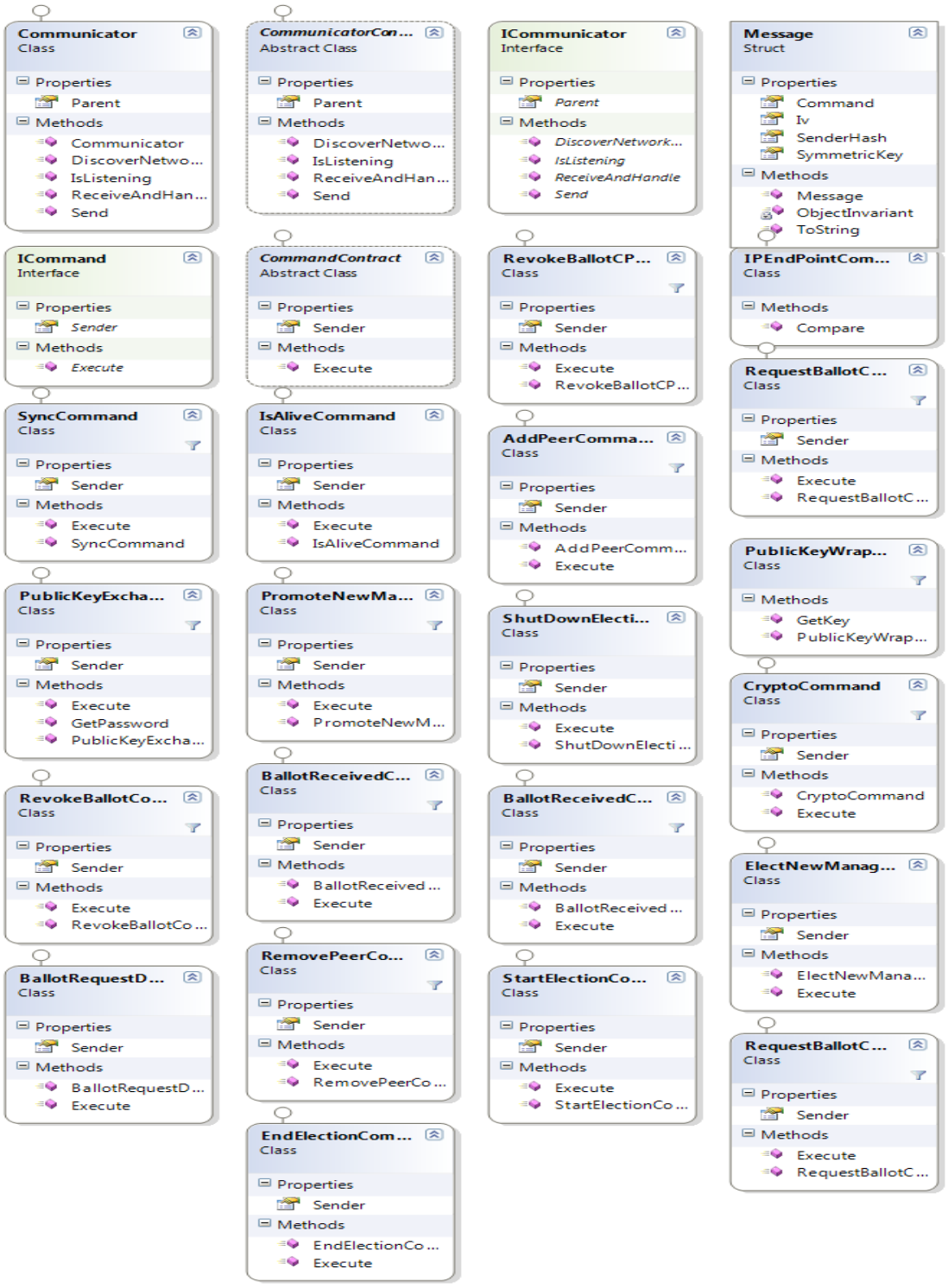
72	During the election, sever the connection to a station	the station is removed from the managers list of peers	Yes	None

17.2 Class diagrams

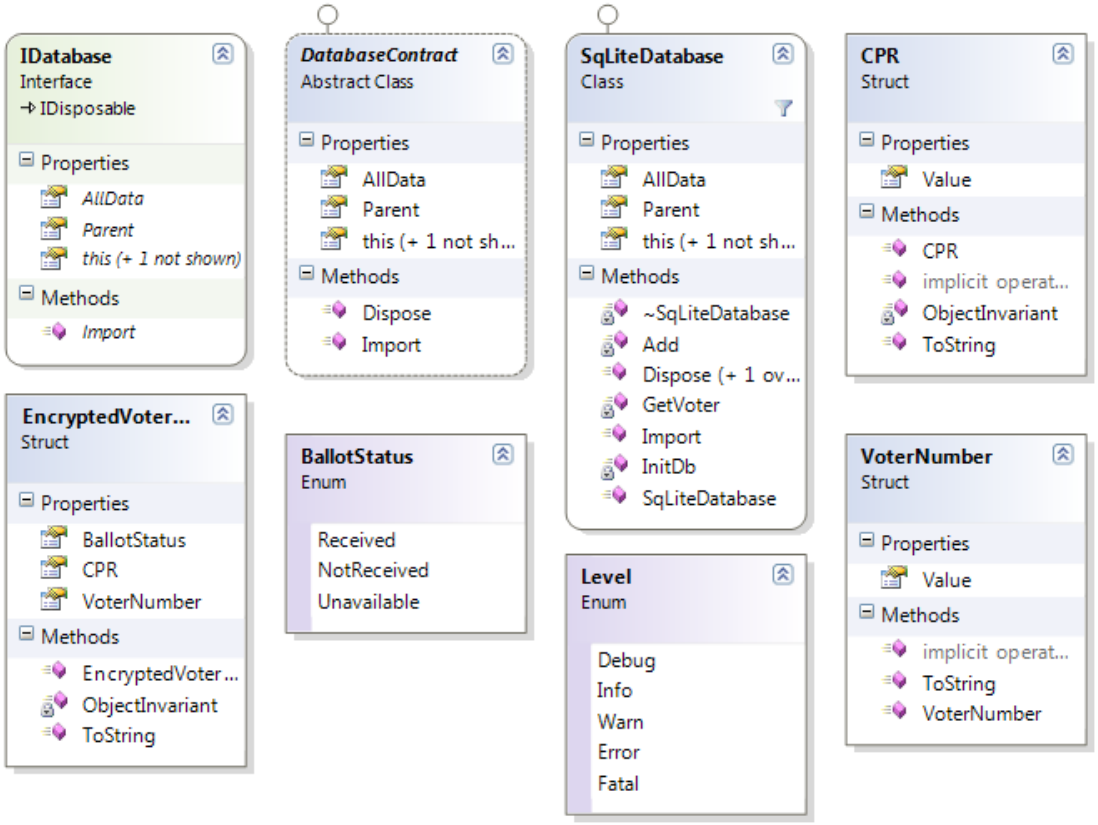
Aegis DVL - All



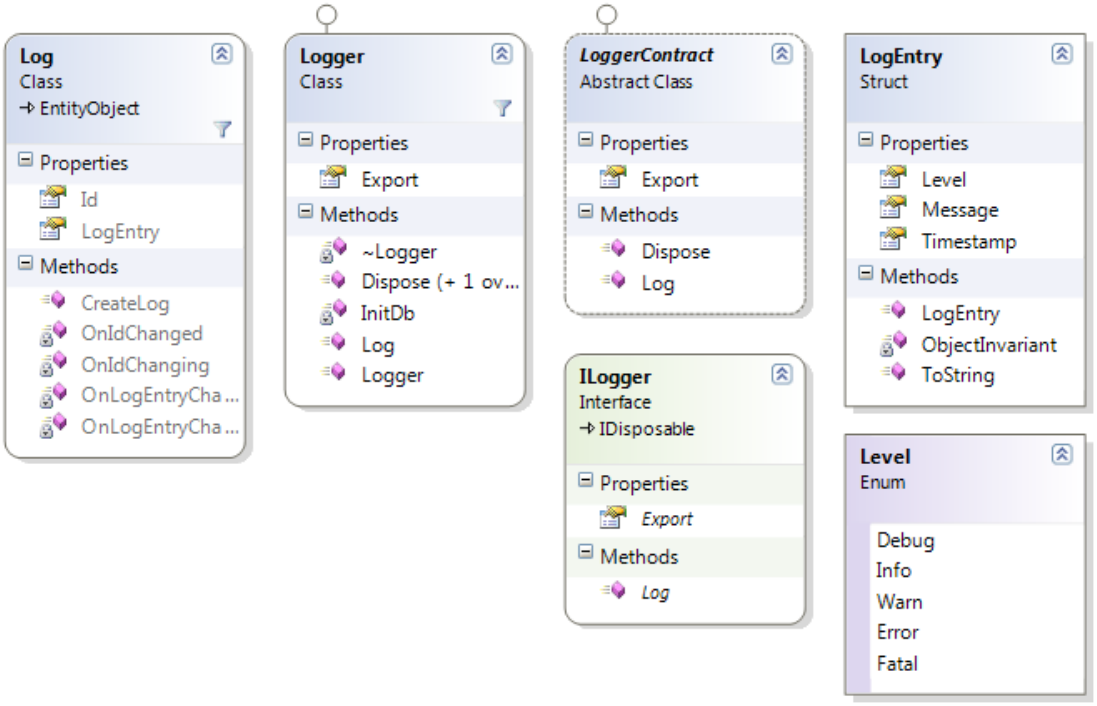
Aegis DVL - Commands and Communication



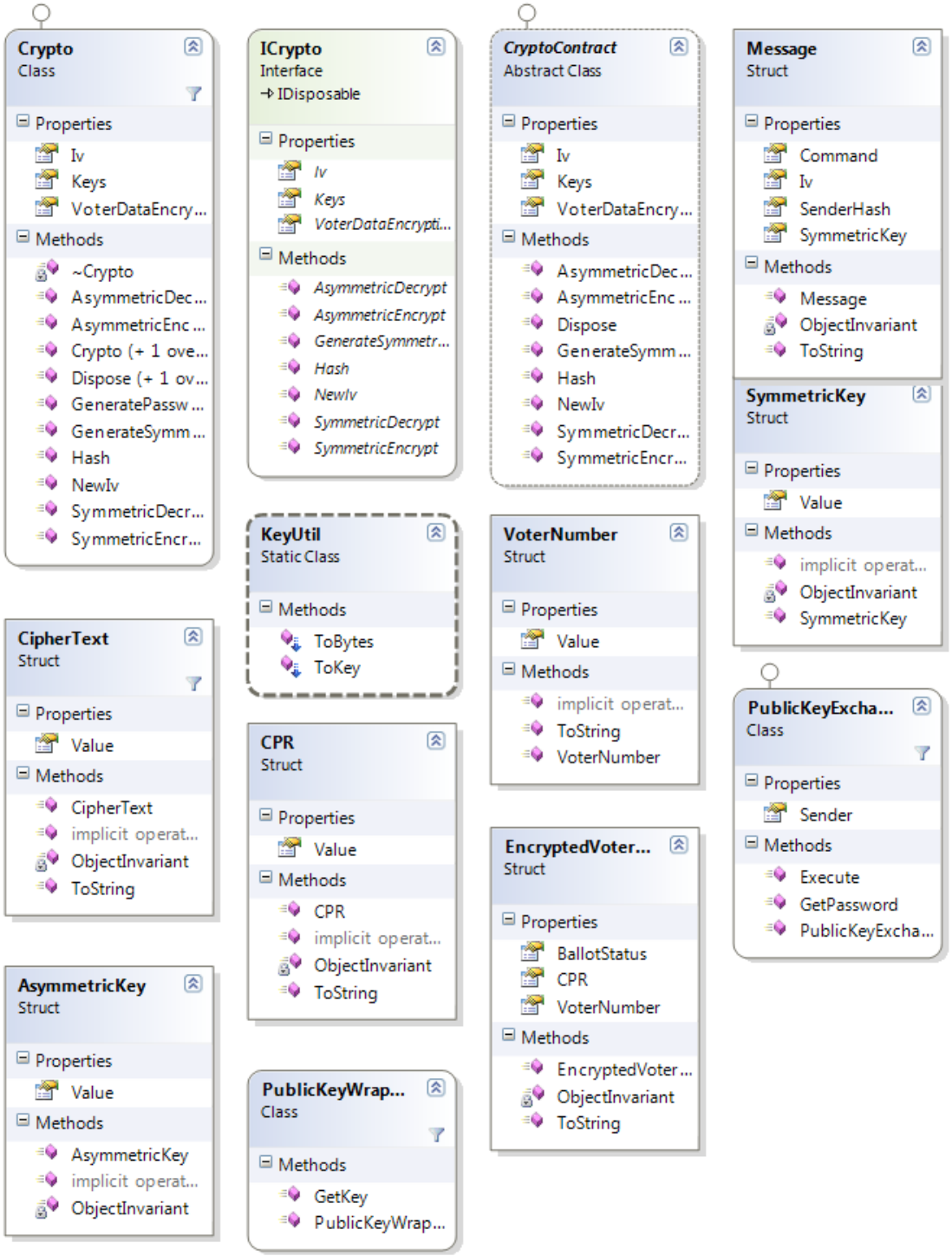
Aegis DVL - Database



Aegis DVL - Logging

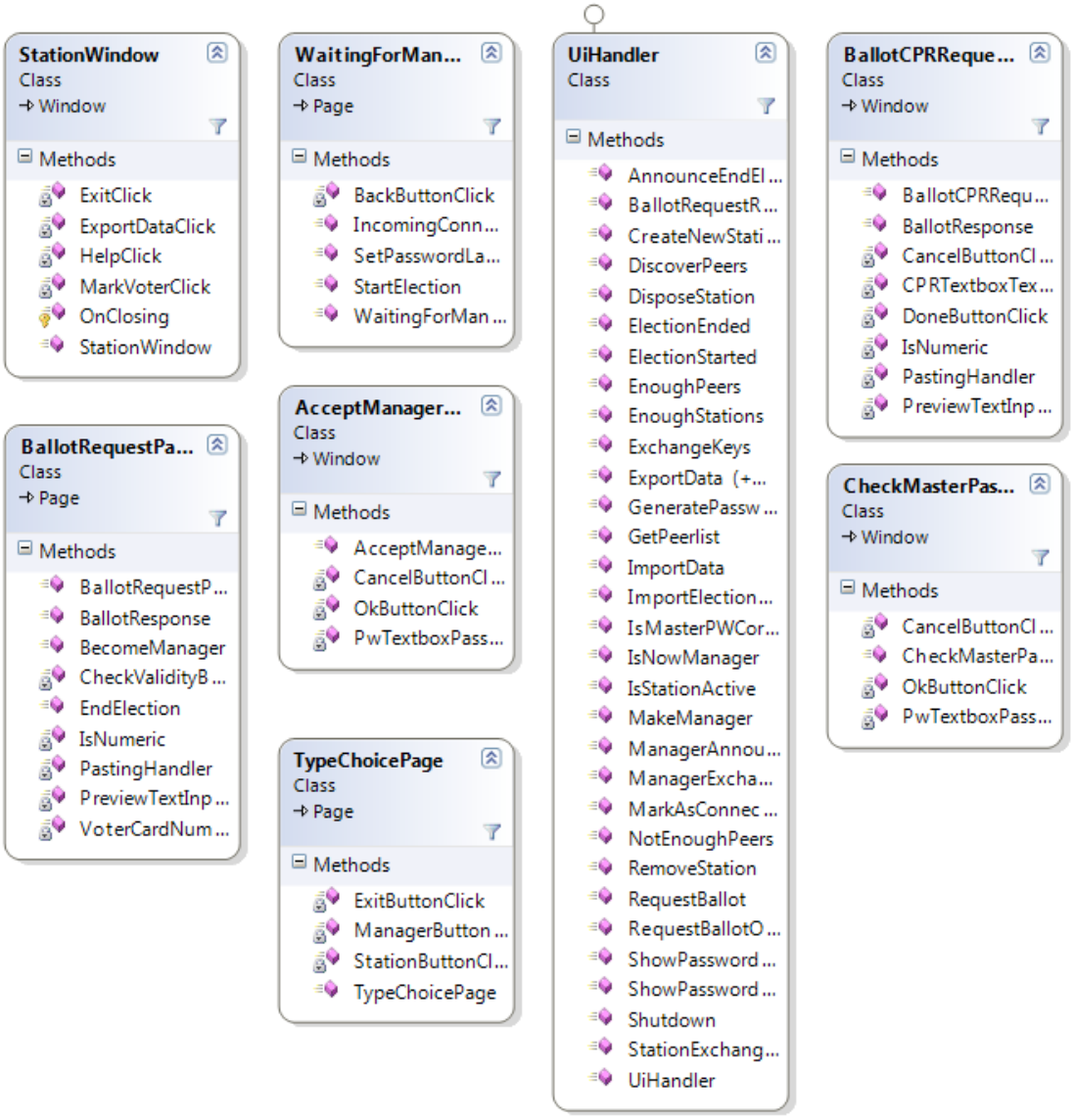


Aegis DVL - Crypto

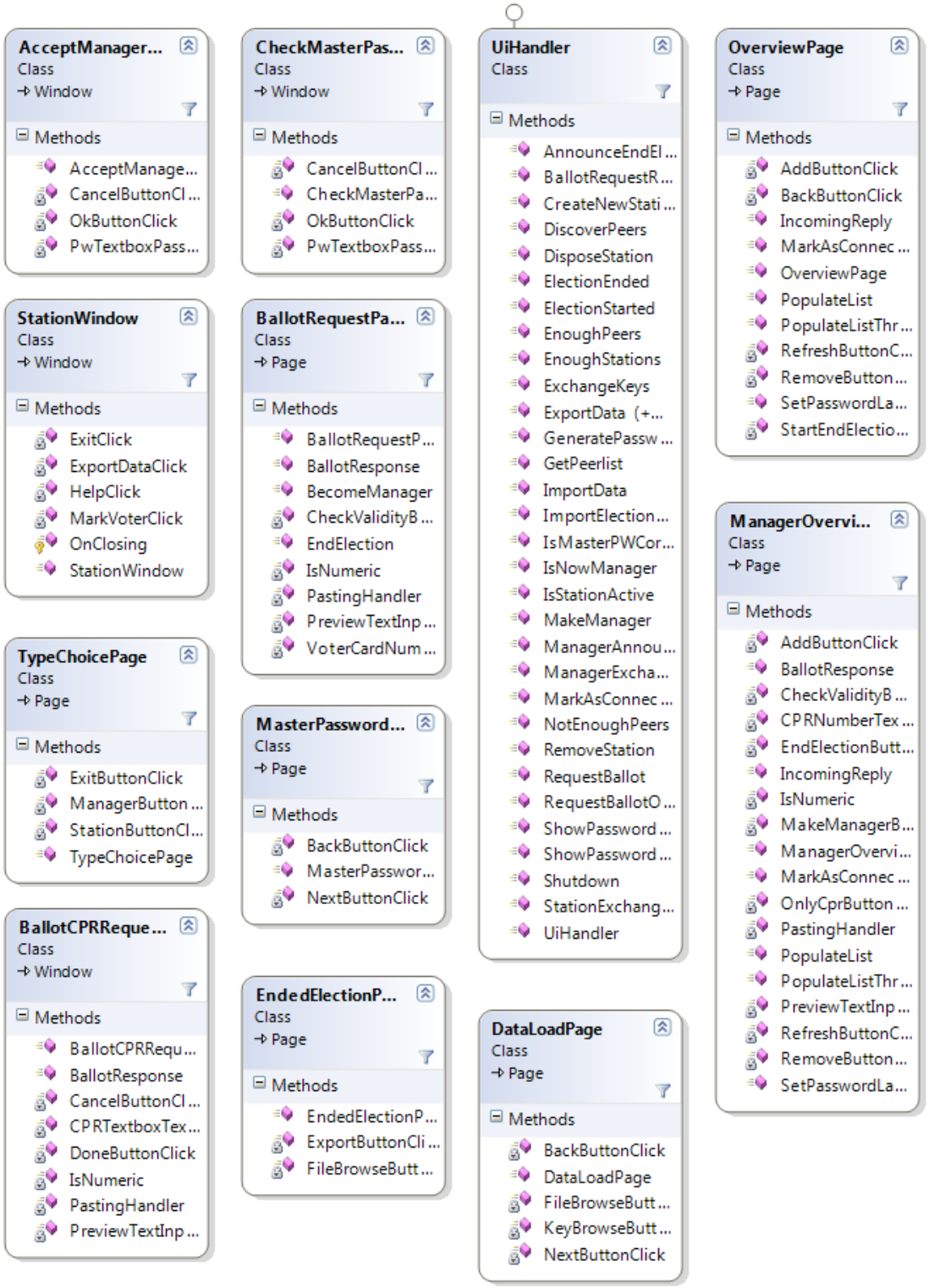


Aegis DVL User interface - All

Aegis DVL User interface - Station



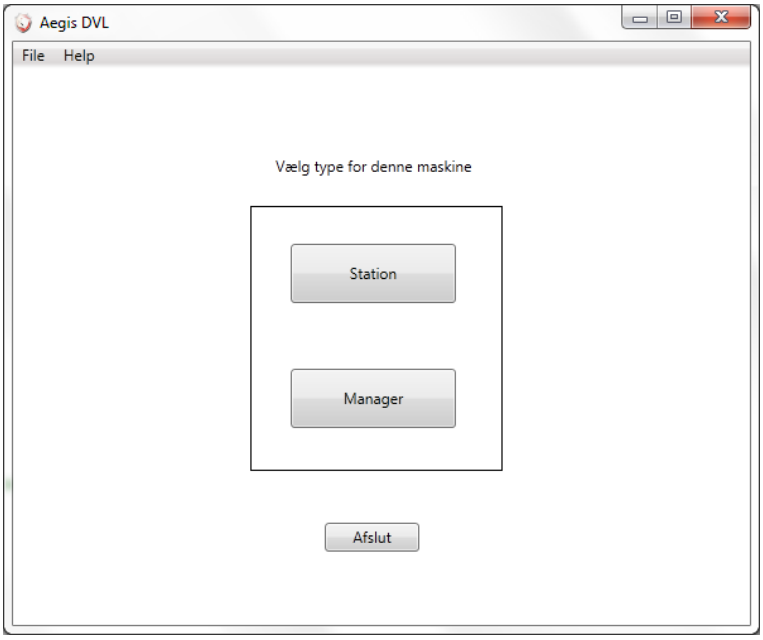
Aegis DVL User interface - Manager



17.3 User manual

Installation

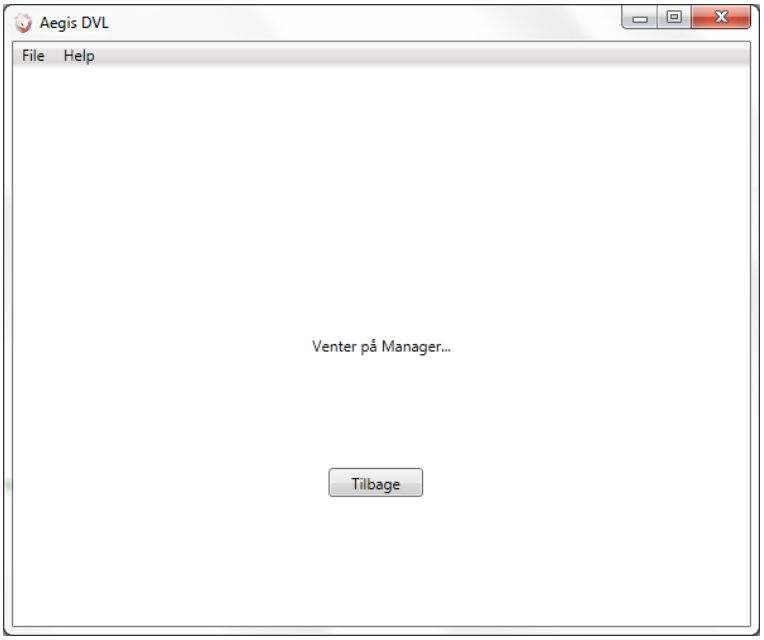
1. Before the election a manager machine should be placed away from the voters and all the station machines should be placed so that they are accessible to the voters.
2. Install the ADO.NET 2.0 Provider for SQLite, (link `http://sourceforge.net/projects/sqlite-dotnet2/`) on each machine. This is the database framework needed to run the program.
3. Install Adobe acrobat reader, (link `http://get.adobe.com/reader/`) or another PDF reader on each machine. The user manual in the program is a PDF file and Adobe acrobat reader is able to display it.
4. Make sure that each machine is in the 192.168.0.1 - 192.168.255.255 IP range.
5. When using this application for the first time Windows will ask you if you want to allow Aegis DVL to pass through your firewall. You need to allow this.
6. Start the Digital Voter List application on each of the machines.
7. You are now presented with this screen:



Choose Manager on the manager machine and Station on all the station machines.

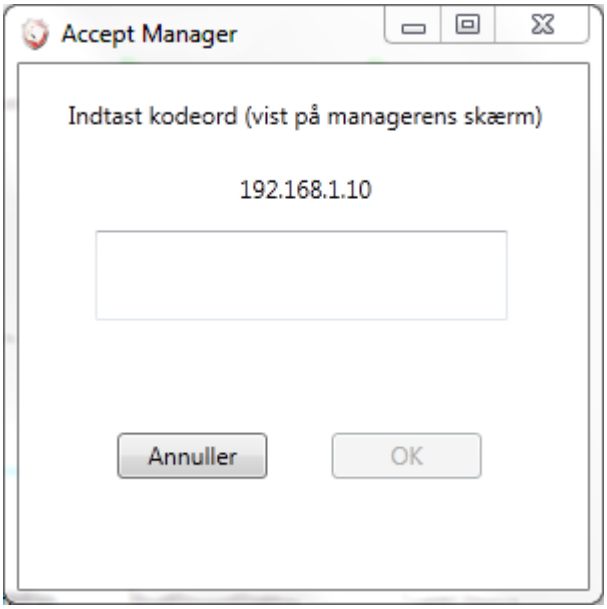
Station usage

1. After you have selected Station you are presented with this page:



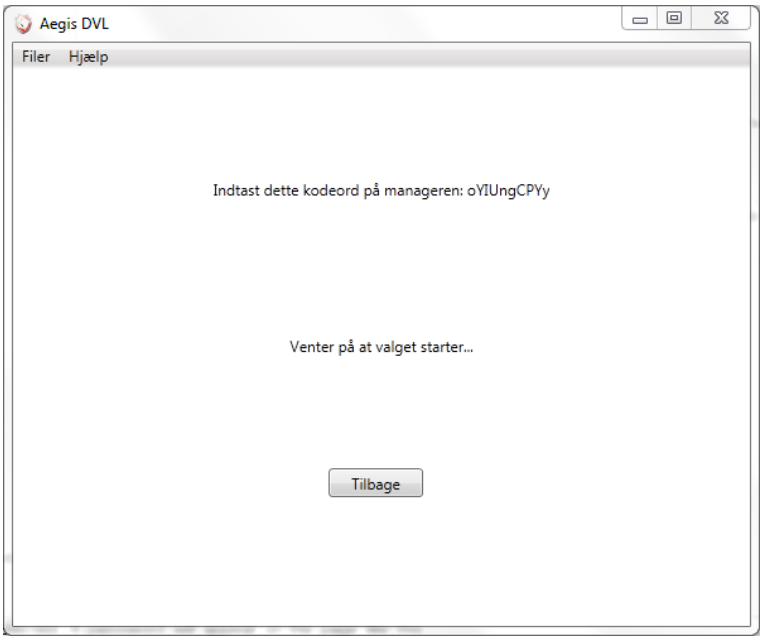
   This screen is displayed until a manager connects.

2. When a manager connects a password is shown on his screen and you are presented with this screen:



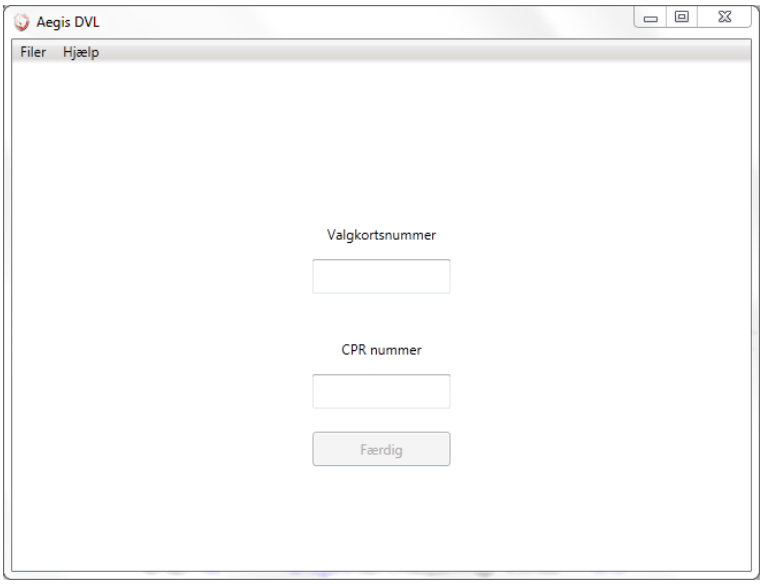
Type the password displayed on the manager in this window and press OK.

3. When the password has been accepted, the reverse process begins. Now a password is displayed on your screen like this:

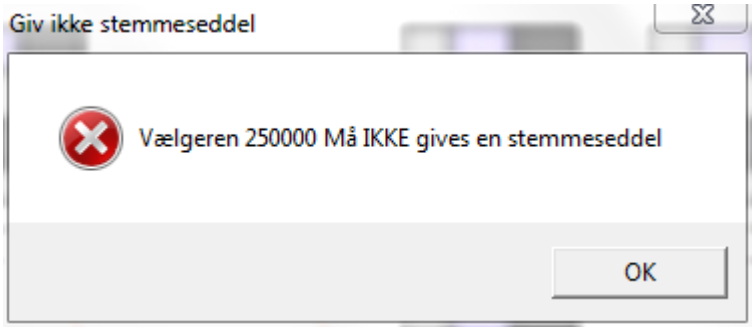


Have the manager type this password in and the text on your screen switches to "Venter på at valget starter" which is displayed until the manager decides to start the election.

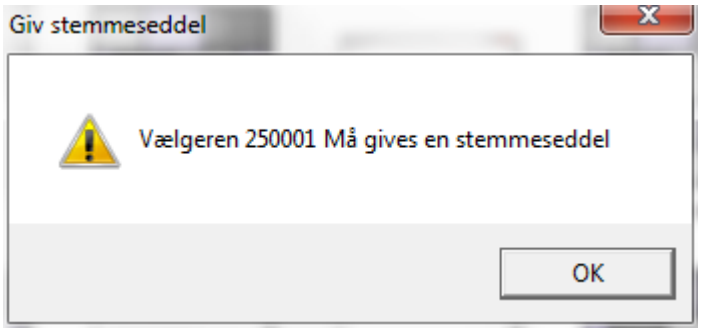
4. When the election starts you are presented with this screen:



From this screen voters can scan/type their voter numbers and type in their CPR numbers. When this is done you can press "Færdig" and one of the following dialogues is shown:



This indicates that the voter is either not eligible to vote at this venue or that he has already been handed a ballot.

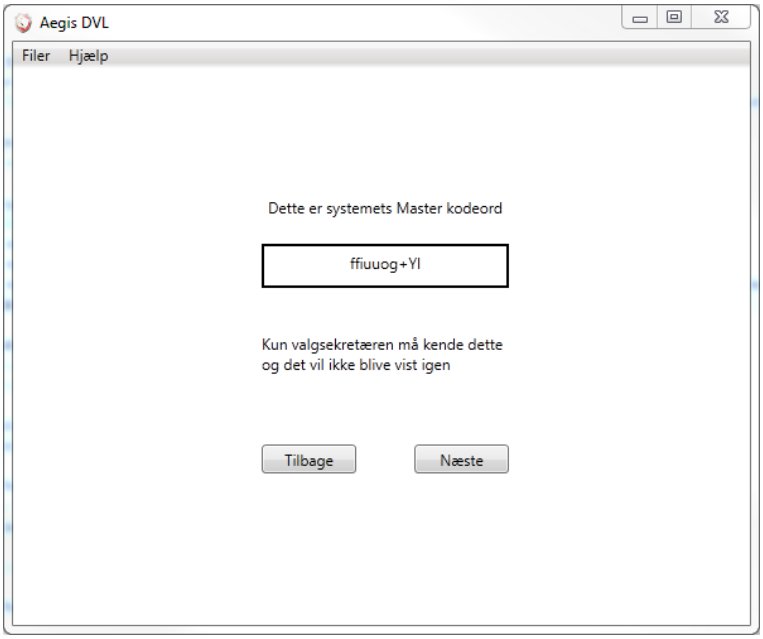


This indicates that the system has accepted the voter number and CPR number and that this voter can now be handed a ballot.

5. This process can be repeated until the manager decides that the election has ended.
6. When the election has ended the application automatically shuts down.
7. When the manager has exported the data and everyone is sure that the election has run as expected it is safe to delete the Voters.data file.

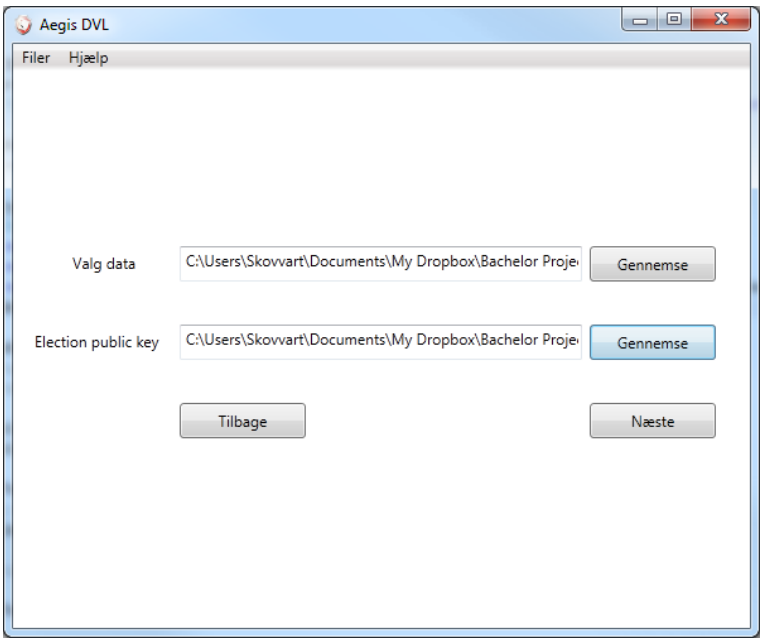
Manager usage

1. After you have selected Manager you are presented with this page:



This window displays the master password. It should only be read by the election secretary and is never shown again! It is used to start an election, end an election, register a voter only with his CPR number and access the log database.

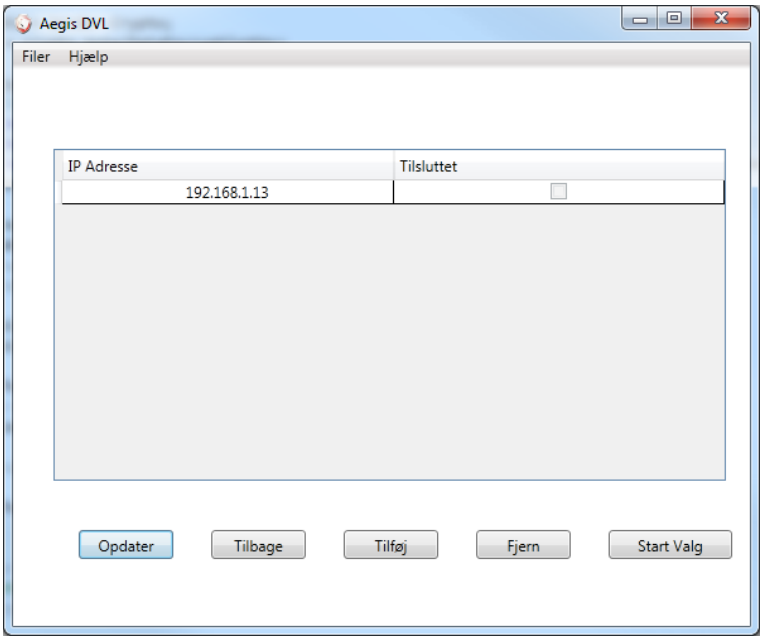
2. When you press "Næste" you are presented with the Data Load Page:



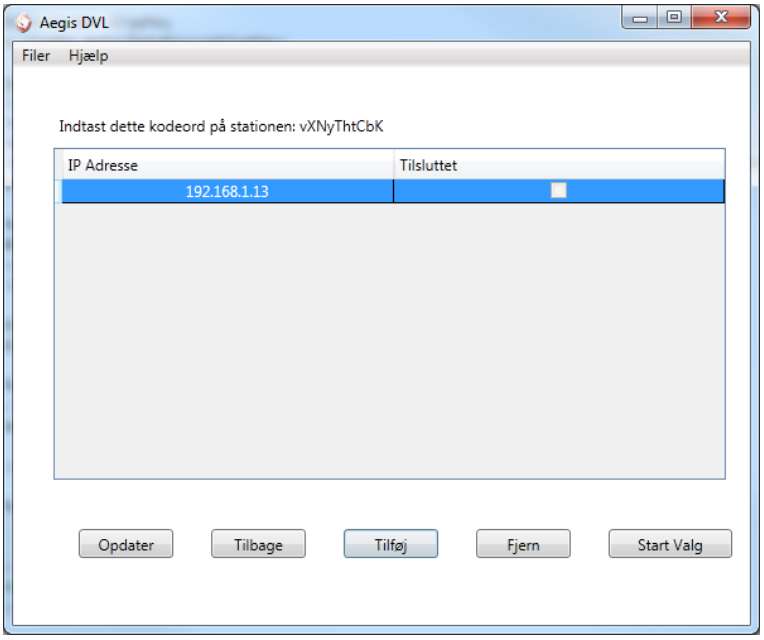
From here you can choose the file location of the voter data the system needs to import

and the encryption key for the voter data in question. When you have found these press "Næste".

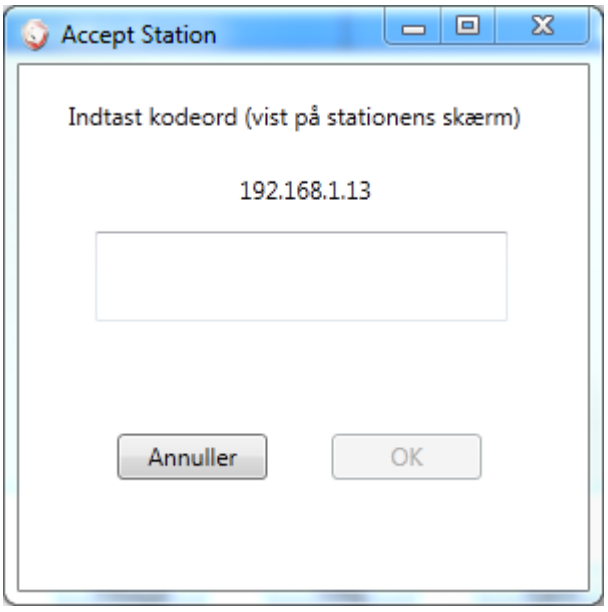
3. You are now presented with this page:



From here you have several options. "Opdater" updates the list of stations you can connect to. "Tilbage" takes you back to the page showing the data loading. It generates a new master password which should be used henceforth. "Tilføj" attempts to connect to the station you have selected. A password appears on the page like this:



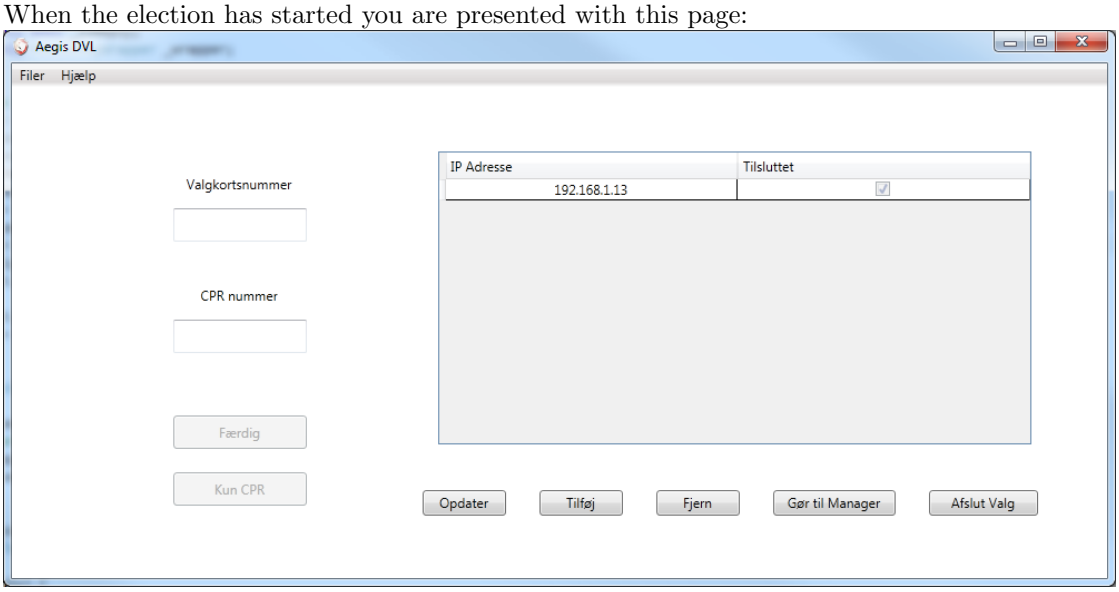
and the station needs to input the password. After the station has entered the password and pressed "OK" you are asked for a password displayed on the station like this:



When you enter the right password the station appears as connected in the list. Pressing "Fjern" removes the stations as a peer, and announces to the remaining peers that they must do the same. A removed peer is ignored. "Start valg" asks you for the master password and start the election like so:

NOTICE: be aware that the system must always have at least four active machines to function. If this is not the case you are not able to start the election.

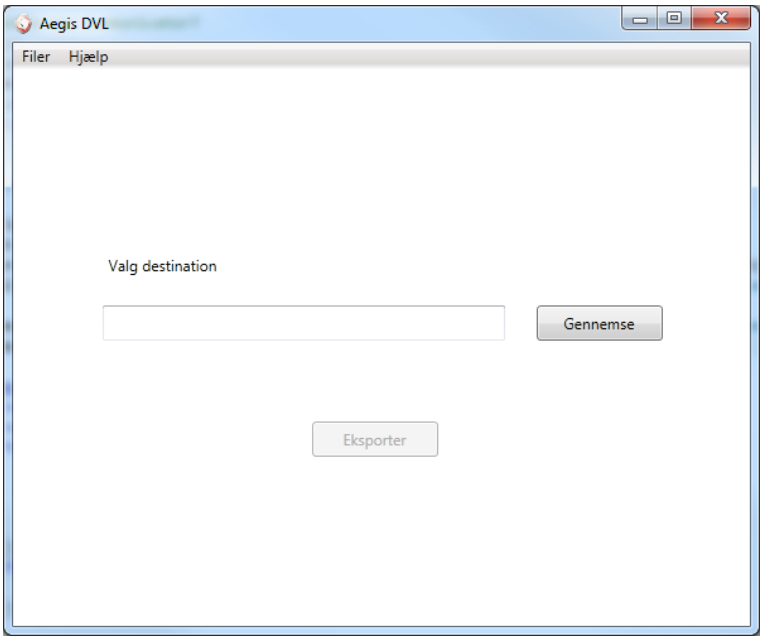
4. When the election has started you are presented with this page:



This page is a combination of the previous page and the voting page from the station. The right side of the page functions exactly like the previous screen and the right side screen gives you the opportunity to mark voters with voter number and CPR number or just the CPR number provided you know the master password.

5. The only difference between the right side of the screen and the previous window is that the "Start Valg" has been replaced by "Afslut Valg" which lets you end the election provided you know the master password. When this is pressed the election ends, the station

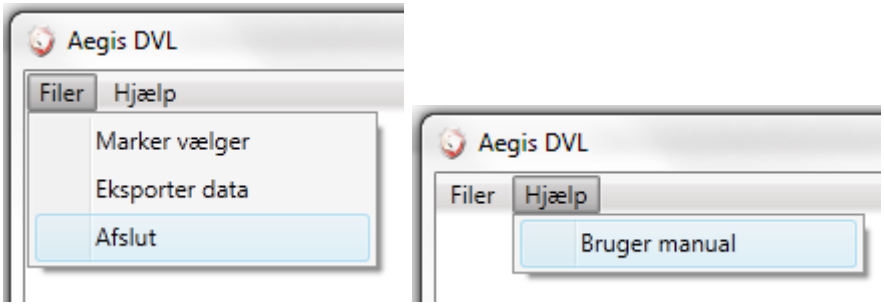
machines closes their applications and you are presented with this page:



6. Here you can export the voter data to a destination of your choice.

Other

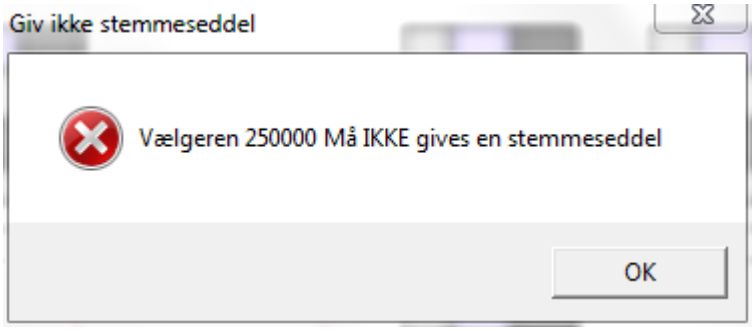
At any time in the program you can choose "Marker vælger", "Eksporter Data" or "Afslut" from the "Filer" menu or "Bruger manual" from the "Hjælp" menu.



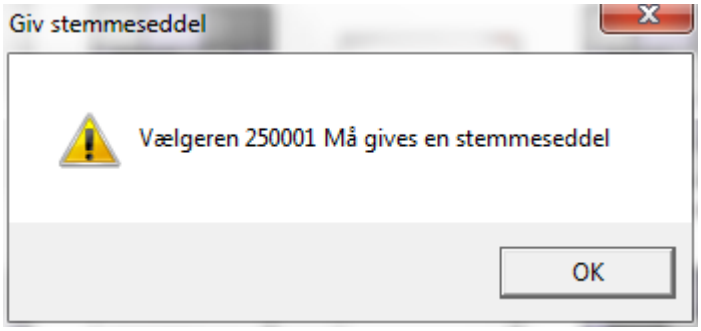
- "Marker vælger" opens this dialog:

Here you can mark a voter with only their CPR number, provided you know the master password. After you have entered the CPR number you are asked to enter the master password in this window:

When this is done you can press "OK" and one of the following dialogues is shown:



This indicates that the voter is either not eligible to vote at this venue or that he has already been handed a ballot.

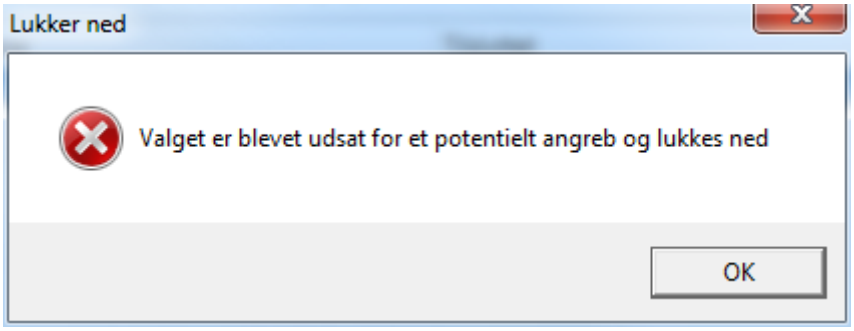


This indicates that the system has accepted the voter number and CPR number and that this voter can now be handed a ballot.

- "Eksporter data" opens a dialog where you choose where to export the voter data. After you have chosen a destination, you are asked to enter the master password. When this is done successfully, the data is exported to the chosen location and the election continues.
- "Afslut" asks you to enter the master password. If entered correctly, the application closes.
- "Bruger manual" opens a PDF file containing this user manual.

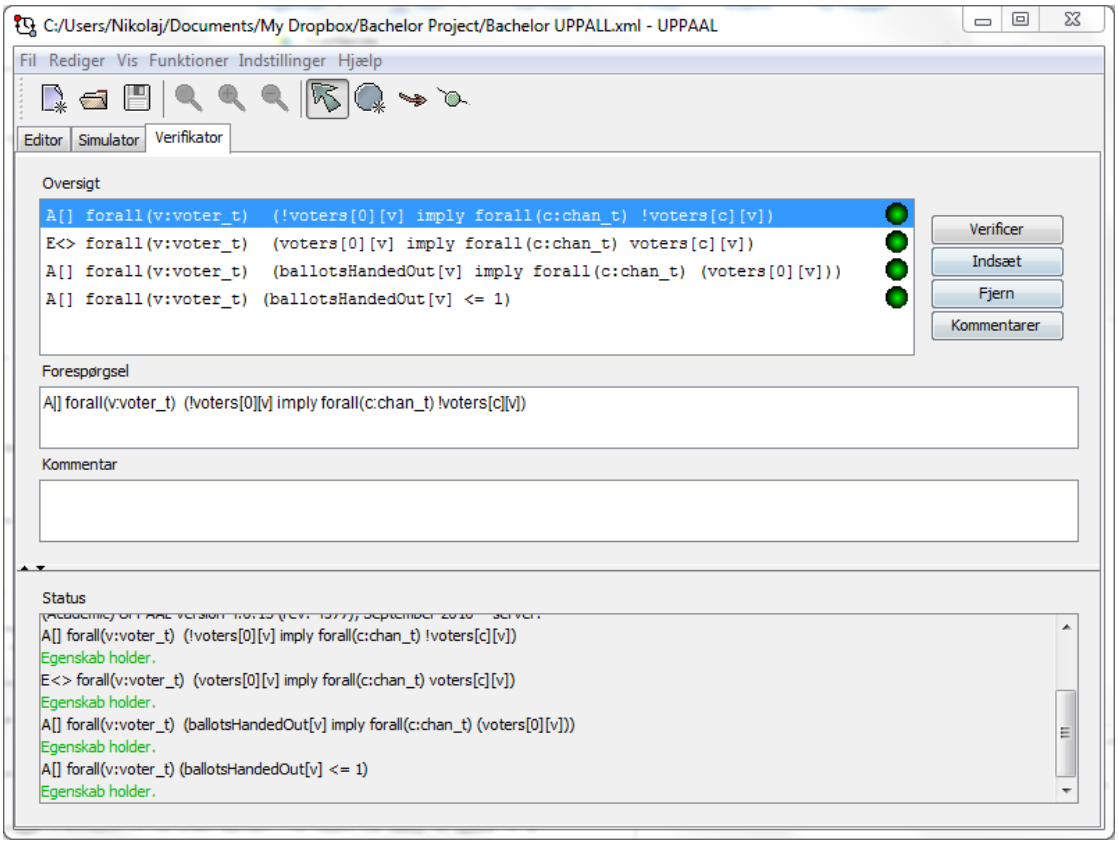
If the manager machine should lose the connection to the network or lose power the remaining stations automatically elects one of the stations as the new manager and the user interface reflects it.

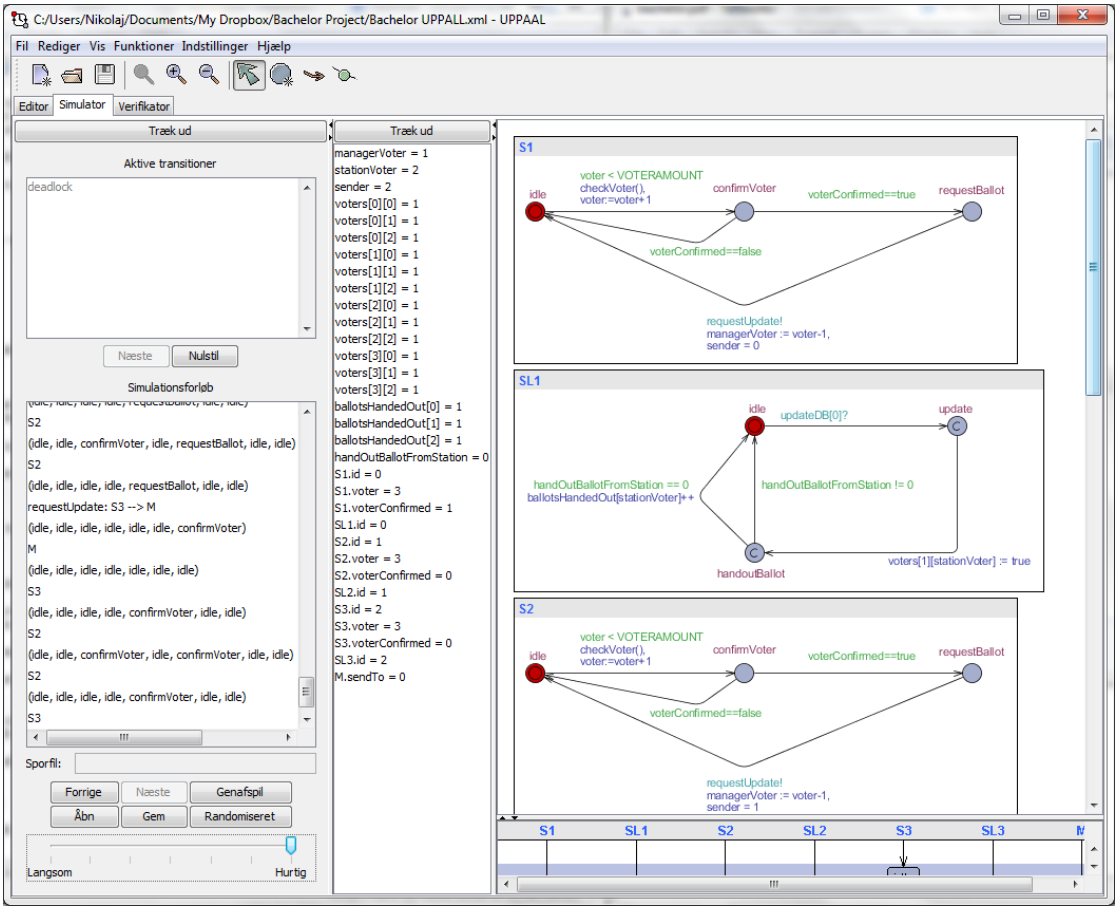
If the election should be a victim of an attack the detection triggers a shutdown of the entire election. This means this dialog appears on all machines:

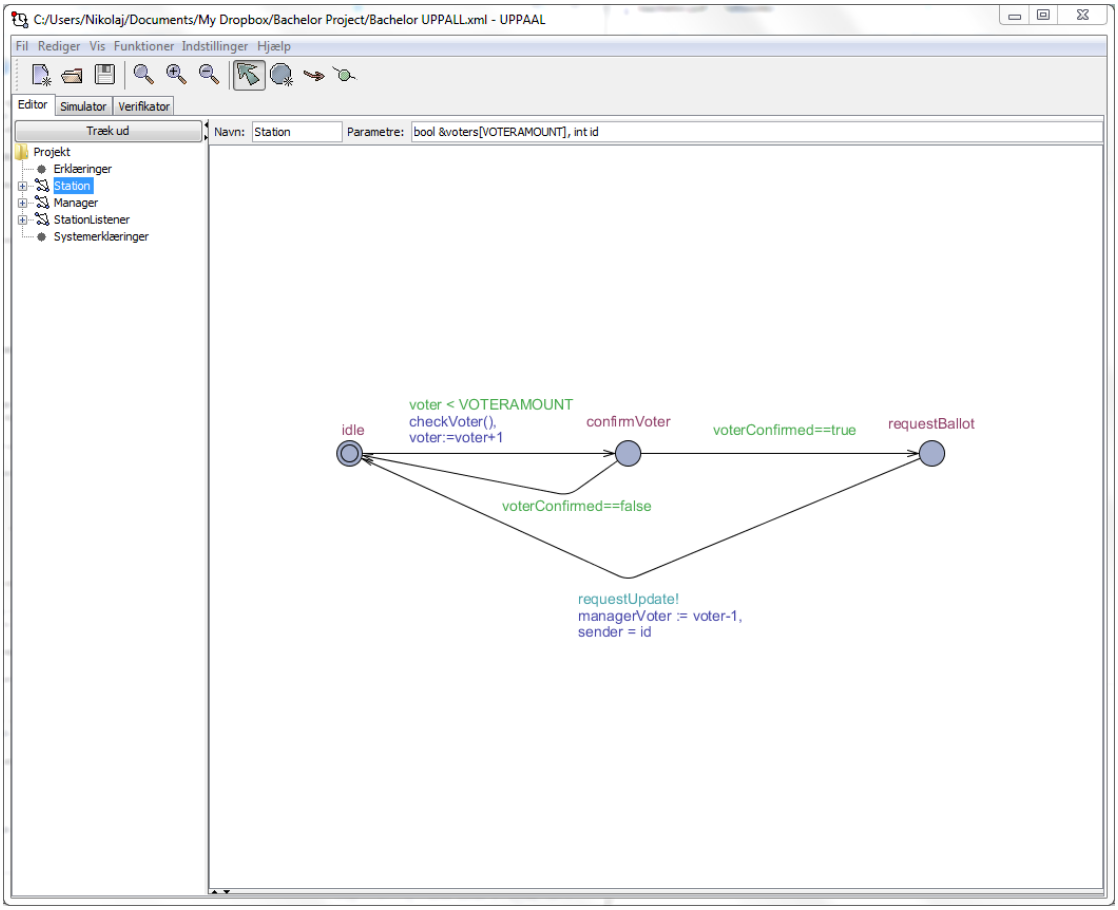


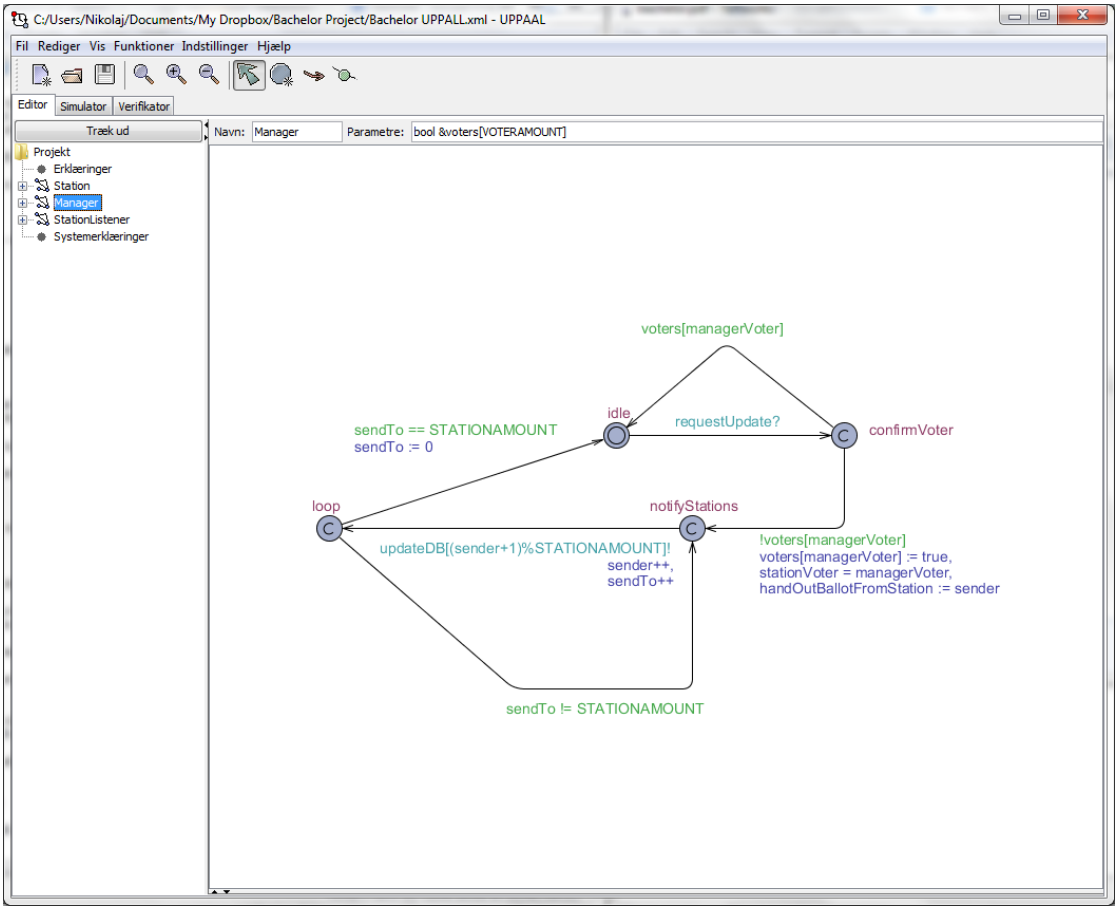
When "OK" is pressed the application closes.

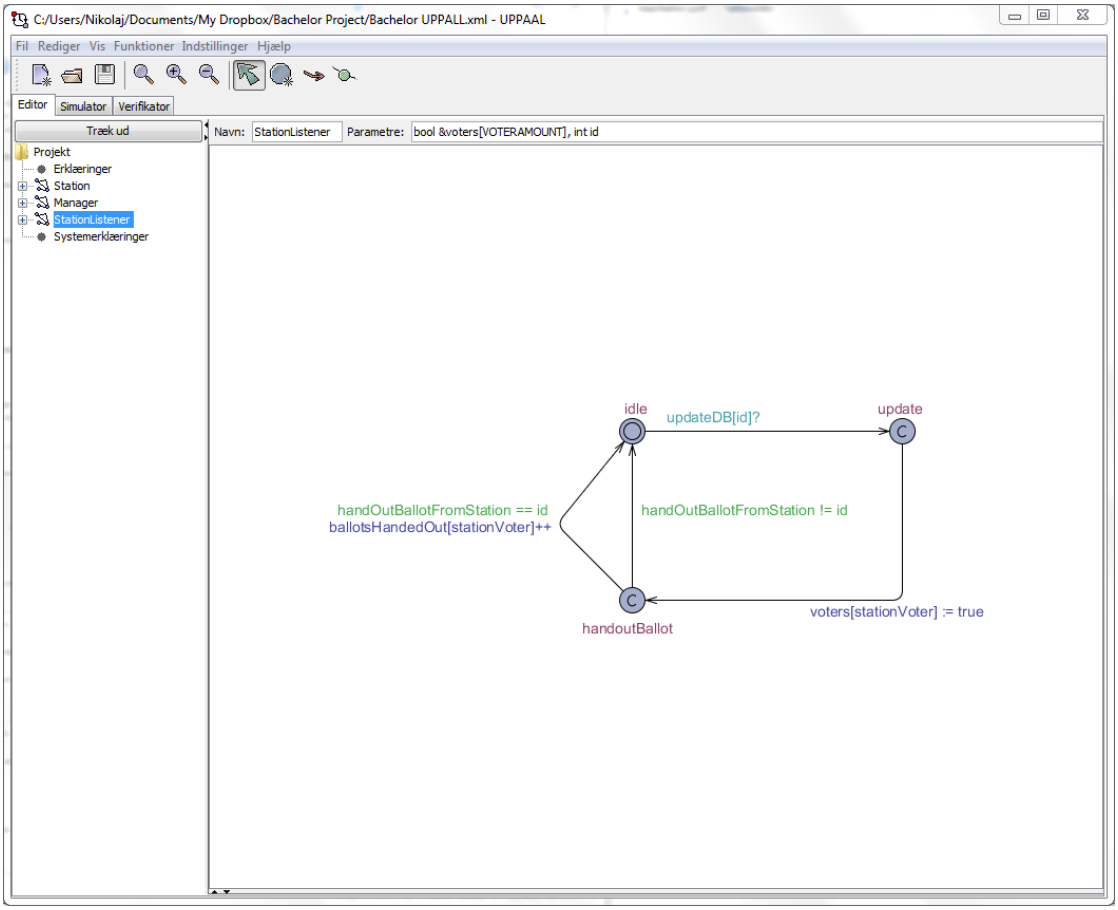
17.4 UPPAAL

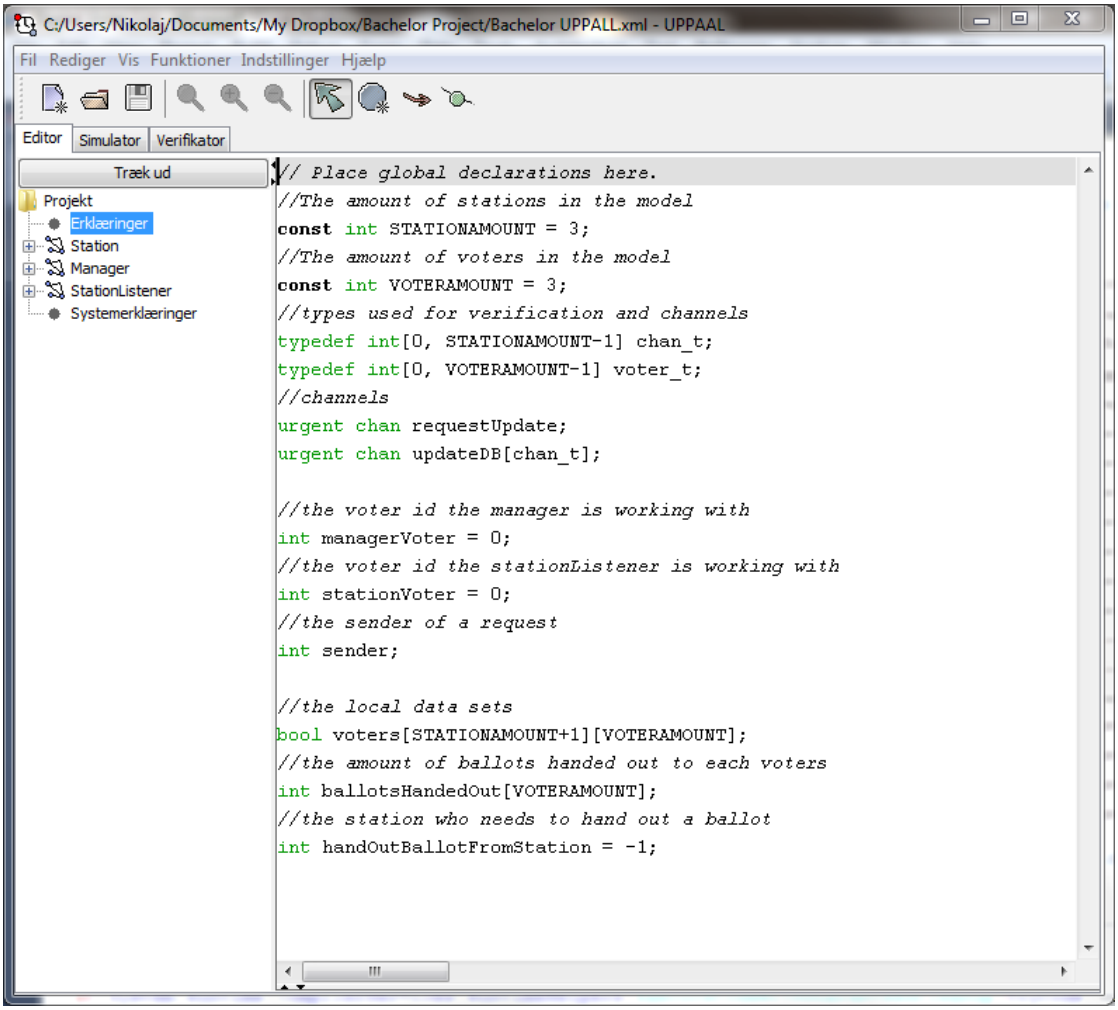


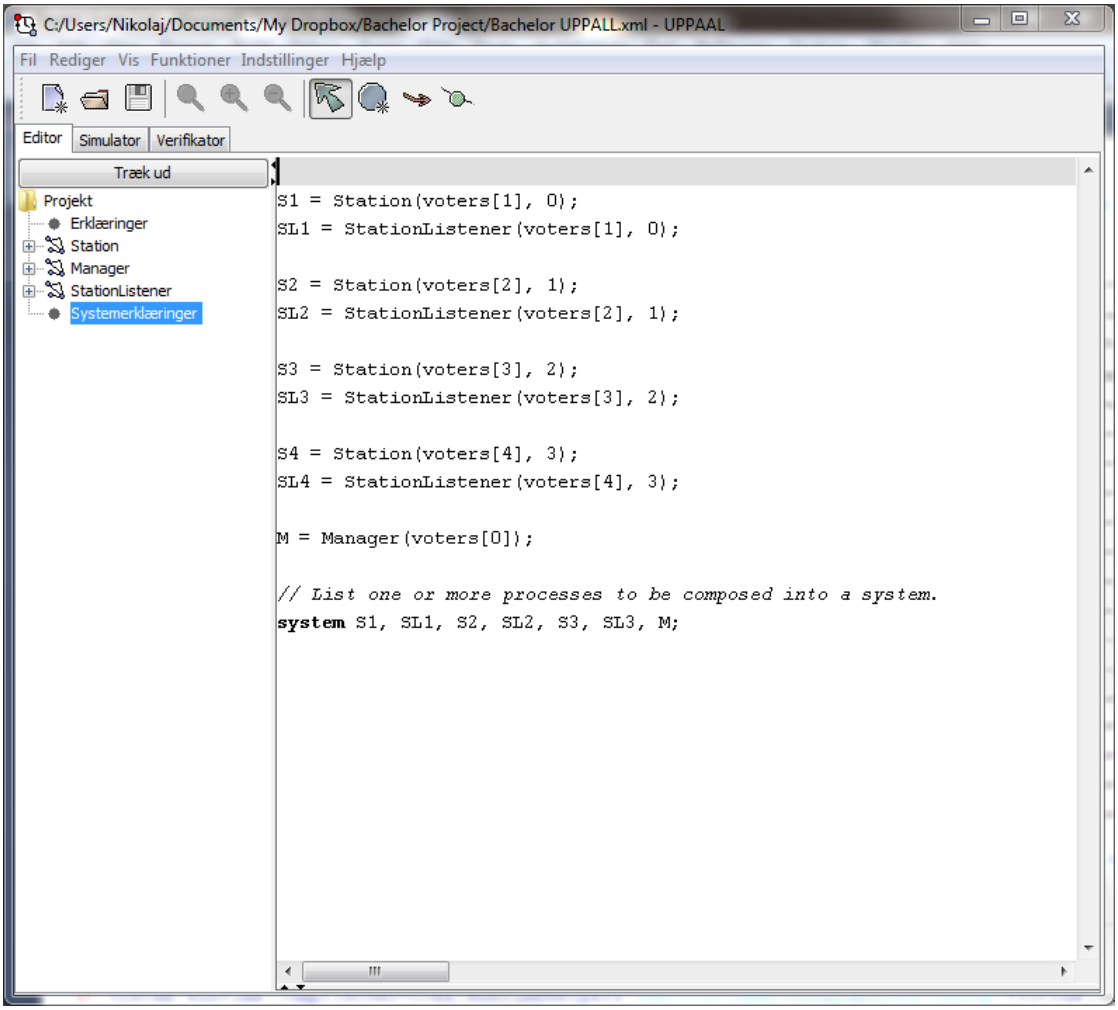

C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Aktive transitioner
deadlock
Næste Nulstil
Simulationsforløb
S2
(idle, idle, confirmVoter, idle, requestBallot, idle, idle)
S2
(idle, idle, idle, idle, requestBallot, idle, idle)
requestUpdate: S3 --> M
(idle, idle, idle, idle, idle, idle, confirmVoter)
M
(idle, idle, idle, idle, idle, idle, idle)
S3
(idle, idle, idle, idle, confirmVoter, idle, idle)
S2
(idle, idle, confirmVoter, idle, confirmVoter, idle, idle)
S2
(idle, idle, idle, idle, confirmVoter, idle, idle)
S3
Sporfil:
Forrige Næste Genafspil
Åbn Gem Randomiseret
Langsom Hurtig
Træk ud
managerVoter = 1
stationVoter = 2
sender = 2
voters[0][0] = 1
voters[0][1] = 1
voters[0][2] = 1
voters[1][0] = 1
voters[1][1] = 1
voters[1][2] = 1
voters[2][0] = 1
voters[2][1] = 1
voters[2][2] = 1
voters[3][0] = 1
voters[3][1] = 1
voters[3][2] = 1
ballotsHandedOut[0] = 1
ballotsHandedOut[1] = 1
ballotsHandedOut[2] = 1
handOutBallotFromStation = 0
S1.id = 0
S1.voter = 3
S1.voterConfirmed = 1
SL1.id = 0
S2.id = 1
S2.voter = 3
S2.voterConfirmed = 0
SL2.id = 1
S3.id = 2
S3.voter = 3
S3.voterConfirmed = 0
SL3.id = 2
M.sendTo = 0
S1
idle
voter < VOTERAMOUNT
checkVoter(),
voter:=voter+1
confirmVoter
voterConfirmed==true
requestBallot
voterConfirmed==false
requestUpdate!
managerVoter := voter-1,
sender = 0
SL1
idle
updateDB[0]?
update
handOutBallotFromStation == 0
ballotsHandedOut[stationVoter]++
handOutBallotFromStation != 0
handoutBallot
voters[1][stationVoter] := true
S2
idle
voter < VOTERAMOUNT
checkVoter(),
voter:=voter+1
confirmVoter
voterConfirmed==true
requestBallot
voterConfirmed==false
requestUpdate!
managerVoter := voter-1,
sender = 1
S1 SL1 S2 SL2 S3 SL3

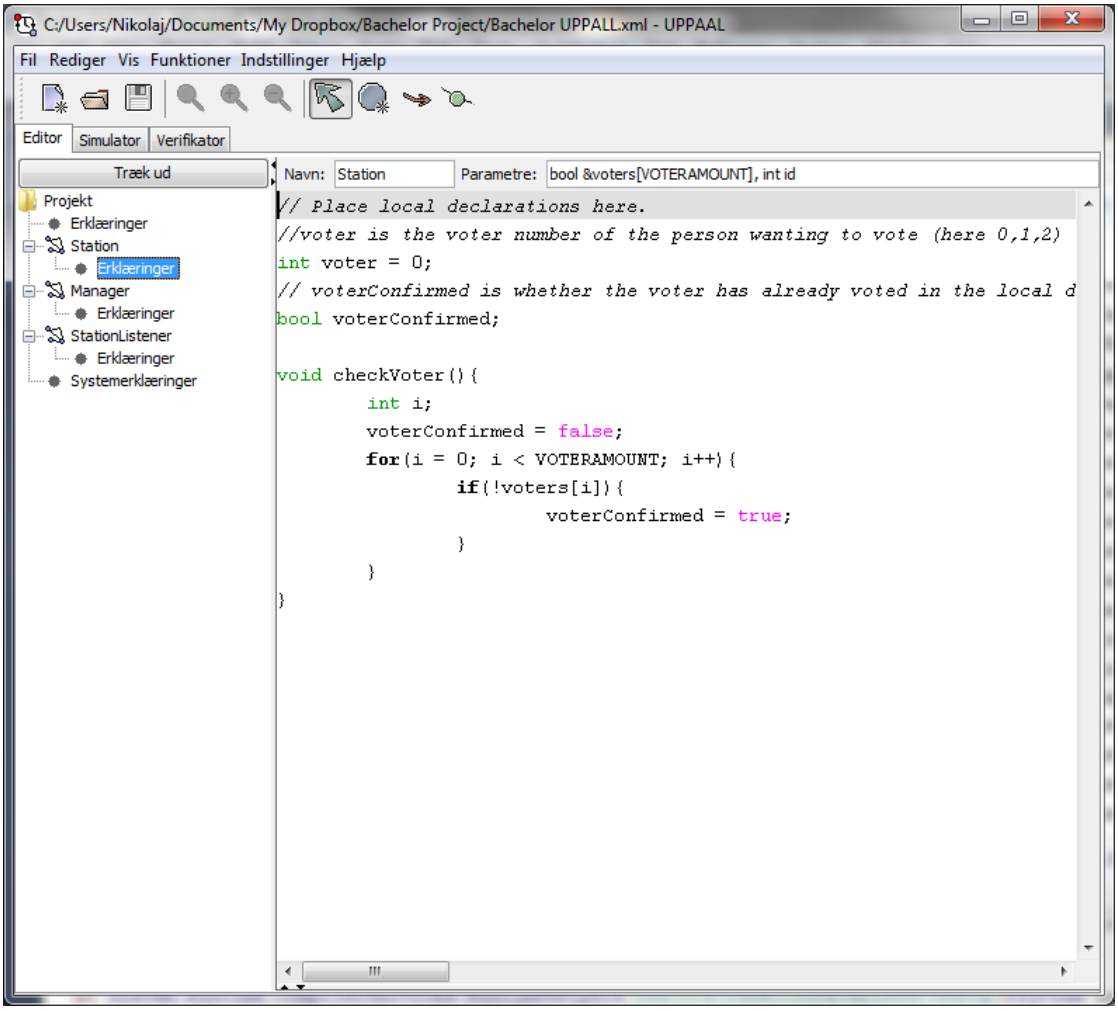

C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Navn: Station
Parametre: bool &voters[VOTERAMOUNT], int id
Projekt
Erklæringer
Station
Manager
StationListener
Systemerklæringer
idle
voter < VOTERAMOUNT
checkVoter(),
voter:=voter+1
confirmVoter
voterConfirmed==true
requestBallot
voterConfirmed==false
requestUpdate!
managerVoter := voter-1,
sender = id

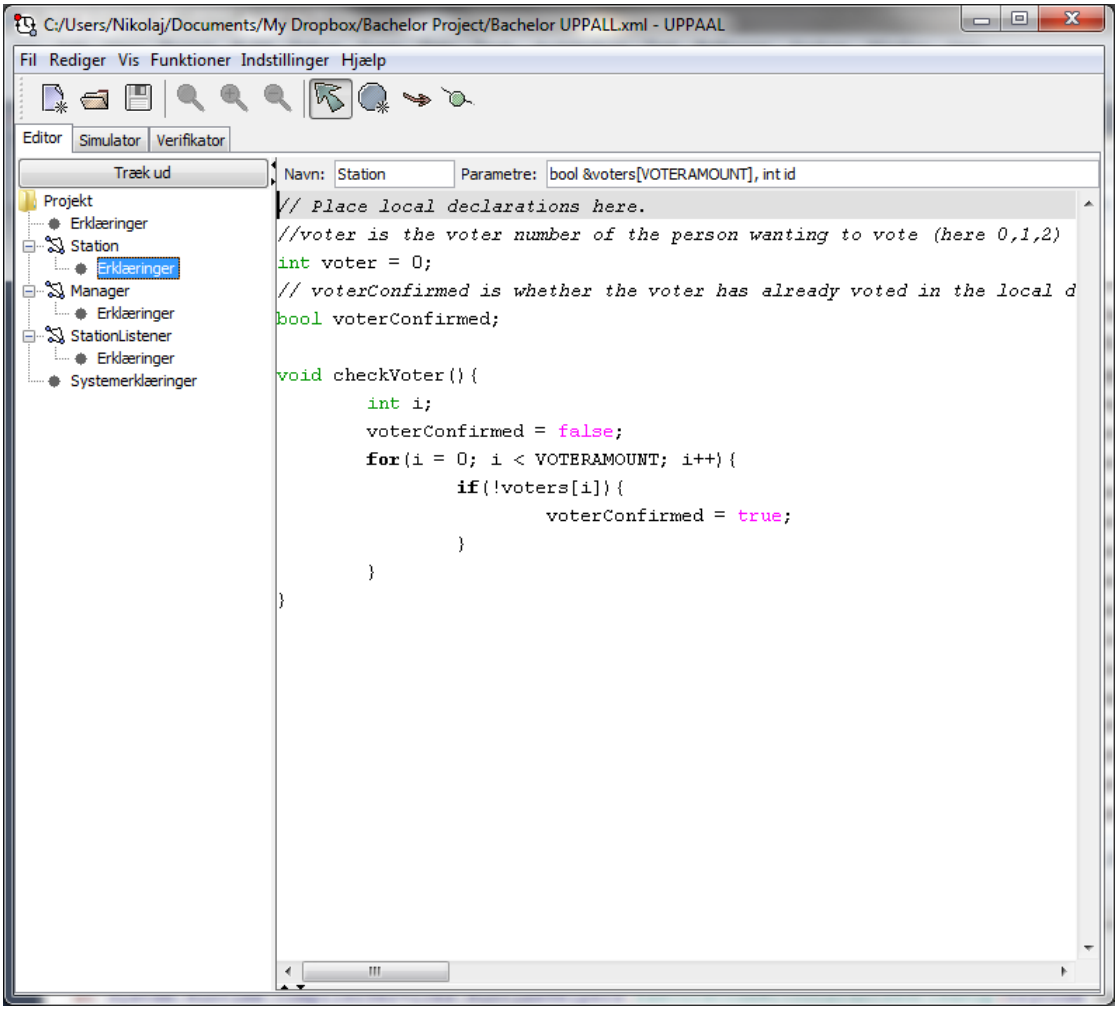

C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Navn: Manager
Parametre: bool &voters[VOTERAMOUNT]
Projekt
Erklæringer
Station
Manager
StationListener
Systemerklæringer
voters[managerVoter]
idle
requestUpdate?
confirmVoter
sendTo == STATIONAMOUNT
sendTo := 0
loop
notifyStations
updateDB[(sender+1)%STATIONAMOUNT]!
sender++,
sendTo++
!voters[managerVoter]
voters[managerVoter] := true,
stationVoter = managerVoter,
handOutBallotFromStation := sender
sendTo != STATIONAMOUNT


C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Navn: StationListener
Parametre: bool &voters[VOTERAMOUNT], int id
Projekt
Erklæringer
Station
Manager
StationListener
Systemerklæringer
idle
updateDB[id]?
update
handOutBallotFromStation == id
ballotsHandedOut[stationVoter]++
handOutBallotFromStation != id
voters[stationVoter] := true
handoutBallot


C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor
Simulator
Verifikator
Træk ud
Projekt
Erklæringer
Station
Manager
StationListener
Systemerklæringer
// Place global declarations here.
//The amount of stations in the model
const int STATIONAMOUNT = 3;
//The amount of voters in the model
const int VOTERAMOUNT = 3;
//types used for verification and channels
typedef int[0, STATIONAMOUNT-1] chan_t;
typedef int[0, VOTERAMOUNT-1] voter_t;
//channels
urgent chan requestUpdate;
urgent chan updateDB[chan_t];
//the voter id the manager is working with
int managerVoter = 0;
//the voter id the stationListener is working with
int stationVoter = 0;
//the sender of a request
int sender;
//the local data sets
bool voters[STATIONAMOUNT+1][VOTERAMOUNT];
//the amount of ballots handed out to each voters
int ballotsHandedOut[VOTERAMOUNT];
//the station who needs to hand out a ballot
int handOutBallotFromStation = -1;


C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor
Simulator
Verifikator
Træk ud
Projekt
Erklæringer
Station
Manager
StationListener
Systemerklæringer
S1 = Station(voters[1], 0);
SL1 = StationListener(voters[1], 0);
S2 = Station(voters[2], 1);
SL2 = StationListener(voters[2], 1);
S3 = Station(voters[3], 2);
SL3 = StationListener(voters[3], 2);
S4 = Station(voters[4], 3);
SL4 = StationListener(voters[4], 3);
M = Manager(voters[0]);
// List one or more processes to be composed into a system.
system S1, SL1, S2, SL2, S3, SL3, M;


C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Projekt
Erklæringer
Station
Erklæringer
Manager
Erklæringer
StationListener
Erklæringer
Systemerklæringer
Navn: Station
Parametre: bool &voters[VOTERAMOUNT], int id
// Place local declarations here.
//voter is the voter number of the person wanting to vote (here 0,1,2)
int voter = 0;
// voterConfirmed is whether the voter has already voted in the local d
bool voterConfirmed;
void checkVoter(){
int i;
voterConfirmed = false;
for(i = 0; i < VOTERAMOUNT; i++){
if(!voters[i]){
voterConfirmed = true;
}
}
}


C:/Users/Nikolaj/Documents/My Dropbox/Bachelor Project/Bachelor UPPALL.xml - UPPAAL
Fil Rediger Vis Funktioner Indstillinger Hjælp
Editor Simulator Verifikator
Træk ud
Projekt
Erklæringer
Station
Erklæringer
Manager
Erklæringer
StationListener
Erklæringer
Systemerklæringer
Navn: Station
Parametre: bool &voters[VOTERAMOUNT], int id
// Place local declarations here.
//voter is the voter number of the person wanting to vote (here 0,1,2)
int voter = 0;
// voterConfirmed is whether the voter has already voted in the local d
bool voterConfirmed;
void checkVoter(){
int i;
voterConfirmed = false;
for(i = 0; i < VOTERAMOUNT; i++){
if(!voters[i]){
voterConfirmed = true;
}
}
}

17.5 Attack trees

Attack trees as described by Schneier in the notation described by Moore et al. with the addition of using <Attack pattern name> to indicate the use of attack patterns in the attack tree. This should make the attacks trees less cluttered and make them easier to investigate. When the notation is used in an attack tree the attack pattern can be substituted in for the identifier. We have also added a parentheses at the end of each action indication the cost of the action in Danish kroner, the number of people required to carry out the action, the technical skill needed to carry out the attack (high, medium or low) and the likelihood of the attack rated from 1 to 5, where 1 is very unlikely and 5 is very likely.

Example:

```
2. Manipulate person(s) responsible for partitioning to manipulate the data
   <Manipulate person(s)>

is equivalent to

2. Manipulate person(s) responsible for partitioning to manipulate the data
     OR     1. Bribe them (20.000/1/low/3)
            2. Force them (0/1/low/4)
            3. Threaten them (0/1/low/4)
```

Attack pattern - Manipulate person(s) (0/1/low/4)
Goal: Force one or more people to do what an attacker wants
Precondition: Targets must be susceptible and the attacker must have the resources necessary
Attack:

OR 1. Bribe them (20.000/1/low/3)
2. Force them (0/1/low/4)
3. Threaten them (0/1/low/4)

Postcondition: The targets will now do what the attacker wants

Attack pattern - Gain access to partitioning machine (0/1/low/4)
Goal: Gain access to the machine where the full data-set of the election is held and is being partitioned for each election venue
Precondition: -
Attack:

OR 1. Be responsible for partitioning (0/1/low/1)
2. Manipulate person(s) responsible for partitioning to manipulate the data
<Manipulate person(s)>
3. Manipulate the data without the person(s) responsible noticing (0/1/medium/1)
4. <Digitally force access>
5. Physically force entry and the attacker manipulating the data (0/1/medium/3)

Postcondition: Attacker now has access to all data on the partitioning machine

Attack pattern - Acquire private key used to decrypt data (0/1/low/4)
Goal: To acquire the private key used to decrypt voter data (such as voter-number, CPR number and ballot status)
Precondition: Attack must know who generates or where the private key is generated
Attack:

OR 1. Be responsible for generating the private key (0/1/medium/1)
2. Manipulate person(s) responsible for generating the private key
<Manipulate person(s)>
3. Steal the private key without being noticed (0/1/medium/1)

Postcondition: The attacker now knows how to decrypt data

Attack pattern - Acquire public key used to encrypt data (0/1/low/4)
Goal: To acquire the public key used to encrypt voter data (such as voter-number, CPR number and ballot status)
Precondition:
Attack:

OR 1. Gain access to a machine and read the public key from RAM (0/1/high/1)
2. Acquire the USB device with the election-venue data (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate person(s) transporting it
<Manipulate person(s)>
3. Be the person responsible for generating the public key (0/1/high/1)

Postcondition: Attacker now knows how to encrypt data

Attack pattern - Digitally force access (0/0/high/2)
Goal: Attacker forces access to the machine through digital means and can execute arbitrary code
Precondition: Attacker must have a computer from which he can control the execution and the skills to do so
Attack:

OR 1. A machine connected to the DVL-machines is available through the internet (0/0/high/1)
2. A malicious machine is attached to the network (0/0/high/1)

3. A DVL-machine is compromised to begin with (0/0/high/2)

Postcondition: Attacker can execute arbitrary code

Attack pattern - Acquire the database key (0/0/high/2)
Goal: Acquire the database password, to grant access to the database
Precondition: The attacker wants to acquire the key used to connect to the local database
Attack:

AND 1. <Digitally force access>
2. Acquire database key from secure memory (0/0/high/2)

Postcondition: The attacker knows the database key and can access the encrypted data

Attack pattern - Impersonate other voters (0/1/high/1)
Goal: Attacker impersonates other voters to gain access to more ballots and therefore more votes
Precondition: The identification proof must be enough to convince the election officials of the identity
Attack:

OR 1. Acquire CPR number and identification-proof (0/1/high/1)
AND 1. Manually request election official to confirm the identity and hand you a ballot (0/1/low/5)
2. Identify CPR and voter-number combinations (0/0/high/1)
OR 1. Acquire voter-cards and CPR number-combination (0/0/high/1)
2. Decrypt database (0/1/high/2)
AND 1. <Acquire private key used to decrypt the data>
2. <Acquire the database-key>
3. Request ballot at station like any other voter (0/1/low/5)

Postcondition: Attacker has access to multiple ballots and is able to vote multiple times

Attack pattern - Access transportation unit and destroy (0/1/low/2)
Goal: To access the unit (e.g. vehicle) which transports the ballots and/or data and destroy it
Precondition: The necessary means to gain access to the transportation unit
Attack:

AND 1. Locate the transportation unit (0/1/low/2)
2. Gain access to transportation unit (0/1/low/4)
3. Destroy (0/1/low/5)

Postcondition: Attacker now has access to the goods inside the transportation unit and can destroy it at will.

Attack pattern - Enter election venue and destroy (0/1/low/4)
Goal: Enter the election venue and destroy physical objects
Precondition: The attacker must know where an election venue is located, and must have the means to destroy the objects
Attack:

AND 1. <Gain access to election venue>
2. Destroy objects (0/1/low/5)

Postcondition: The objects are destroyed, and must be replaced for the election to proceed

Attack pattern - Gain access to election venue (0/1/low/4)
Goal: To gain access to the election venue
Precondition: Attacker must know the location of the election venue
Attack:

OR 1. Physically force access (0/1/low/4)
2. Steal key (0/1/medium/3)
3. Be an insider (0/1/medium/1)
4. Manipulate an insider
<Manipulate person(s)>

Postcondition: Attacker has access to the election venue

Tree 1

To tamper with the election for personal benefit (0/1/low/4)

OR 1. Manipulate the digital data (0/1/low/4)

OR 1. Before the election (0/1/low/4)

OR 1. During partitioning

<Gain access to partitioning machine>

2. During transportation to election venue (0/1/high/4)

OR 1. Exchange the USB device (0/1/high/4)

AND 1. Physically acquire the device (0/1/low/4)

OR 1. Steal without people transporting it noticing (0/1/low/1)

2. Manipulate people transporting it

<Manipulate person(s)>

2. <Acquire public key used to encrypt the data>

3. Encrypt tampered data-set with public key (0/1/high/5)

4. Write data to own USB device (0/1/low/5)

5. Give new USB device to people transporting it (0/1/low/5)

2. Manipulate the data on the existing USB device (0/1/high/4)

AND 1. Physically acquire the device (0/1/low/4)

OR 1. Steal without people transporting it noticing (0/1/low/1)

2. Manipulate people transporting it

<Manipulate person(s)>

2. Replace or manipulate (0/1/high/4)

OR 1. Manipulate (0/1/high/4)

AND 1. <Acquire private key used to decrypt the data>

2. <Acquire public key used to encrypt the data>

3. Decrypt data-set (0/0/high/5)

4. Manipulate data (0/0/high/5)

5. Encrypt tampered data-set with public key (0/0/high/5)

6. Write data to USB device (0/0/low/5)

2. Replace (0/1/high/4)

AND 1. <Acquire public key used to encrypt the data>

2. Encrypt tampered data-set with public key (0/0/high/5)

3. Write data to USB device (0/0/low/5)

3. On manager-machine before election has started (0/1/high/4)

AND 1. Gain access to the manager-machine (0/1/low/4)

OR 1. Be the election official(s) (0/1/medium/1)

2. Force access (0/1/low/4)

OR 1. Physically force access (0/1/low/3)

2. Digitally force access

<Digitally force access>

3. Force an insider to grant access

<Manipulate person(s)>

2. Replace or manipulate (0/1/high/4)

OR 1. Manipulate (0/1/high/4)

AND 1. <Acquire private key used to decrypt the data>

2. <Acquire public key used to encrypt the data>

3. Decrypt data-set (0/0/high/5)

4. Manipulate data (0/0/high/5)

5. Encrypt tampered data-set with public key (0/0/high/5)
6. Replace data (0/1/low/5)

2. Replace (0/1/high/4)
AND 1. <Acquire public key used to encrypt the data>
2. Encrypt tampered data-set with public key (0/0/high/5)
3. Replace data (0/1/low/5)

2. During the election (0/1/low/4)
OR 1. Manipulate the database on all the machines (0/1/medium/4)
AND 1. Gain access to all machines (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
2. <Acquire public key used to encrypt the data>
3. <Acquire the database key>
4. Manipulate or add records to the database (0/1/medium/5)

2. Gain access to multiple ballots by continuously revoking ballot-received (0/1/low/4)
AND 1. Gain access to the management machine (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
OR 1. Manipulate person with access to the manager-machine
<Manipulate person(s)>
2. Digitally force access
<Digitally force access>
2. Gain access to all signatures and keys, and broadcast revoke-commands to all stations (0/1/high/1)

3. Prevent people from voting by marking them as having received a ballot (0/1/low/1)
AND 1. Identify CPR and voter-number combinations (0/1/low/5)
OR 1. Acquire voter-cards and CPR numbers (0/1/low/5)
2. Decrypt database (0/1/low/4)
AND 1. <Acquire private key used to decrypt the data>
2. <Acquire the database-key>
2. Mark voters (0/1/low/1)
OR 1. Gain access to machine(s) (0/1/low/1)
OR 1. The management machine and manually mark voters as having received ballots (0/1/medium/1)
2. The station and manually request ballots (0/1/low/1)
2. Update database (0/1/high/1)
AND 1. Obtain public key (0/1/high/1)
2. Obtain database-key (0/1/high/1)
3. Update the database (0/1/low/5)

4. Impersonate other voters
<Impersonate other voters>

3. After the election (0/1/high/4)
OR 1. Before being exported (0/1/high/4)
AND 1. Gain access to the manager-machine (0/1/low/4)
OR 1. Be the election official(s) (0/1/medium/1)
2. Force access (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
3. Force an insider to grant access
<Manipulate person(s)>

2. Replace or manipulate (0/1/high/4)
OR 1. Manipulate (0/1/high/4)
AND 1. <Acquire private key used to decrypt the data>
2. <Acquire public key used to encrypt the data>
3. Decrypt data-set (0/0/high/5)
4. Manipulate data (0/0/high/5)
5. Encrypt tampered data-set with public key (0/0/high/5)
6. Replace data (0/1/high/5)
2. Replace (0/1/high/4)
AND 1. <Acquire public key used to encrypt the data>
2. Encrypt tampered data-set with public key (0/0/high/5)
3. Replace data (0/1/high/5)
2. During transportation (0/1/high/4)
OR 1. Exchange the USB device (0/1/high/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. <Acquire public key used to encrypt the data>
3. Encrypt tampered data-set with public key (0/0/high/5)
4. Write data to own USB device (0/1/high/5)
5. Give new USB device to people transporting it (0/1/low/5)
2. Manipulate the data on the existing USB device (0/1/high/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. Replace or manipulate (0/1/high/4)
OR 1. Manipulate (0/1/high/4)
AND 1. <Acquire private key used to decrypt the data>
2. <Acquire public key used to encrypt the data>
3. Decrypt data-set (0/0/high/5)
4. Manipulate data (0/0/high/5)
5. Encrypt tampered data-set with public key (0/0/high/5)
6. Write data to USB device (0/1/medium/5)
2. Replace (0/1/high/4)
AND 1. <Acquire public key used to encrypt the data>
2. Encrypt tampered data-set with public key (0/0/high/5)
3. Write data to USB device (0/1/medium/5)
3. At the tallying location (0/1/low/4)
OR 1. Be responsible for tallying (0/1/medium/1)
2. Manipulate person(s) responsible for tallying to manipulate the data
<Manipulate person(s)>
3. Manipulate the data without the person(s) responsible noticing (0/1/low/4)
4. <Digitally force access>
5. Physically force entry and the attacker manipulating the data (0/1/low/3)
2. Vote several times without manipulating the digital data (0/1/low/4)
AND 1. Physically gain access to ballots (0/1/low/4)

2. Force election officials to accept them
<Manipulate person(s)>

Tree 2

To destroy the election (0/1/low/4)

OR 1. Physically destroy the storage units when being transported (0/1/low/2)
OR 1. Before the election
<Access transportation unit and destroy>
2. After the election
<Access transportation unit and destroy>
2. Destroy the election stations (0/1/low/4)
OR 1. Before the election
<Enter election venue and destroy>
2. During the election
<Enter election venue and destroy>
3. Destroying ballots (0/1/low/4)
OR 1. Before election (0/1/low/4)
OR 1. When being transported to election venue
<Access transportation unit and destroy>
2. At the election venue (0/1/low/4)
AND 1. <Gain access to election venue>
2. Destroy ballots (0/1/low/5)
2. During the election
<Enter election venue and destroy>
3. After the election (0/1/low/4)
OR 1. At the election venue
<Enter election venue and destroy>
2. During transportation
<Access transportation unit and destroy>
3. At tallying place (0/1/low/3)
AND 1. Locate tallying place (0/1/low/3)
2. Gain access to tallying place (0/1/low/4)
3. Destroy (0/1/low/5)
4. Prevent people from voting at the election venue (0/1/low/2)
OR 1. Prevent them from receiving voter cards (0/1/low/2)
2. Physically prevent them from entering election venue (0/1/low/2)
5. Deleting data (0/1/low/4)
OR 1. Before the election (0/1/low/4)
OR 1. During partitioning
<Gain access to partitioning machine>
2. During transportation to election venue (0/1/medium/4)
OR 1. Delete data on the USB device (0/1/medium/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. Delete the data (0/1/medium/5)
3. (Optional) Give the USB device to people transporting it (0/1/low/5)
3. On manager-machine before election has started (0/1/medium/4)

AND 1. Gain access to the manager-machine (0/1/low/4)
OR 1. Be the election official(s) (0/1/medium/1)
2. Force access (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
3. Force an insider to grant access
<Manipulate person(s)>
2. Delete the data (0/1/medium/5)
2. During the election (0/1/high/2)
OR 1. Delete the database on all the machines (0/1/high/2)
AND 1. Gain access to all machines (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
2. Delete the database (0/1/high/2)
3. After the election (0/1/low/4)
OR 1. Before being exported (0/1/high/2)
AND 1. Gain access to the manager-machine (0/1/low/4)
OR 1. Be the election official(s) (0/1/medium/1)
2. Force access (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
3. Force an insider to grant access
<Manipulate person(s)>
2. Delete the database (0/1/high/2)
2. During transportation (0/1/high/4)
OR 1. Delete data on the USB device (0/1/high/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. Delete the data (0/1/high/5)
3. (Optional) Give the USB device to people transporting it
(0/1/low/5)
3. At the tallying location (0/1/low/4)
OR 1. Be responsible for tallying (0/1/low/1)
2. Manipulate person(s) responsible for tallying to delete the data
<Manipulate person(s)>
3. Delete the data without the person(s) responsible noticing (0/1/high/1)
4. <Digitally force access>
5. Physically force entry and the attacker deleting the data (0/1/low/4)
6. Corrupting data (0/1/low/4)
OR 1. Before the election (0/1/low/4)
OR 1. During partitioning
<Gain access to partitioning machine>
2. During transportation to election venue (0/1/low/4)
OR 1. Corrupt the USB device (0/1/low/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. Corrupt the data (0/1/high/5)
3. (Optional) Give the USB device to people transporting it

(0/1/low/5)

3. On manager-machine before election has started (0/1/high/4)
   AND 1. Gain access to the manager-machine (0/1/low/4)
      OR 1. Be the election official(s) (0/1/medium/1)
         2. Force access (0/1/low/4)
            OR 1. Physically force access (0/1/low/4)
               2. Digitally force access
                  <Digitally force access>
               3. Force an insider to grant access
                  <Manipulate person(s)>
      2. Corrupt the data (0/1/high/5)

2. During the election (0/1/high/4)
   OR 1. Corrupt the database on all the machines (0/1/high/4)
      AND 1. Gain access to all machines (0/1/low/4)
         OR 1. Physically force access (0/1/low/4)
            2. Digitally force access
               <Digitally force access>
      2. Corrupt the data (0/1/high/5)

3. After the election (0/1/low/4)
   OR 1. Before being exported (0/1/high/4)
      AND 1. Gain access to the manager-machine (0/1/low/4)
         OR 1. Be the election official(s) (0/1/low/1)
            2. Force access (0/1/low/4)
               OR 1. Physically force access (0/1/low/4)
                  2. Digitally force access
                     <Digitally force access>
                  3. Force an insider to grant access
                     <Manipulate person(s)>
      2. Corrupt the data (0/1/high/5)
   2. During transportation (0/1/low/4)
      OR 1. Corrupt the USB device (0/1/low/4)
         AND 1. Physically acquire the device
            OR 1. Steal without people transporting it noticing (0/1/low/1)
               2. Manipulate people transporting it
                  <Manipulate person(s)>
         2. Corrupt the data (0/1/high/5)
         3. (Optional) Give the USB device to people transporting it (0/1/low/5)
   3. At the tallying location (0/1/low/4)
      OR 1. Be responsible for tallying (0/1/low/1)
         2. Manipulate person(s) responsible for tallying to corrupt the data
            <Manipulate person(s)>
         3. Corrupt the data without the person(s) responsible noticing (0/1/high/1)
         4. <Digitally force access>
         5. Physically force entry and the attacker corrupting the data (0/1/low/4)

Tree 3

To gain knowledge about a protected part of the election (0/1/low/4)

OR 1. Get access to the digital data before it's partitioned
      <Gain access to partitioning machine>
   2. Gain access to the partitioned data while it's being transported to the election venue (0/1/high/4)
      OR 1. Access the USB device (0/1/high/4)

AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. <Acquire private key used to decrypt the data>
3. Decrypt and read data (0/1/high/5)

3. Physically spy on the voters during the election (0/1/low/1)
OR 1. Place cameras in the election booths (20.000/1/high/1)
AND 1. Locate the election venue and booths (0/1/low/4)
2. Acquire cameras (20.000/1/low/5)
3. Gain access to the election venue
<Gain access to the election venue>
4. Install the cameras in the election booths without anyone noticing (0/1/high/1)
2. Physically be in the election booth to spy (0/1/low/1)

4. Gain access to the digital data during the election (0/1/low/4)
OR 1. Access a database on a machine (0/1/low/4)
AND 1. Gain access to the machine (0/1/low/4)
OR 1. Physically force access (0/1/low/4)
2. Digitally force access
<Digitally force access>
2. <Acquire private key used to decrypt the data>
3. <Acquire the database key>
4. Decrypt and read the data (0/1/high/5)

5. Gain access to the digital data after the election has ended (0/1/low/4)
OR 1. At election venue (0/1/low/4)
Same as gain access to the digital data during the election
2. Intercept the transportation of the exported data (0/1/low/4)
OR 1. Access the USB device (0/1/low/4)
AND 1. Physically acquire the device (0/1/low/4)
OR 1. Steal without people transporting it noticing (0/1/low/1)
2. Manipulate people transporting it
<Manipulate person(s)>
2. <Acquire private key used to decrypt the data>
3. Decrypt and read data (0/1/high/5)
3. At the tallying place (0/1/low/4)
OR 1. Be responsible for tallying (0/1/low/1)
2. Manipulate person(s) responsible for tallying to manipulate the data
<Manipulate person(s)>
3. Manipulate the data without the person(s) responsible noticing
4. <Digitally force access>
5. Physically force entry and the attacker manipulating the data (0/1/low/4)

17.6 Revision history

```
Revision: 334
Author: Skovvart
Date: 2012-05-18 20:06
Message: Increased RSA key-strength. This significantly slows down the constructor-speed.
----
Revision: 333
Author: Skovvart
Date: 2012-05-18 13:59
Message: Optimized a contract.
----
Revision: 332
Author: Skovvart
Date: 2012-05-17 23:43
Message: Added a lock on logger to prevent some threading issues in the tests..
----
Revision: 331
Author: Skovvart
Date: 2012-05-17 19:55
Message: Fixed a comment in station, made logger commit every entry every time again
----
Revision: 330
Author: Skovvart
Date: 2012-05-17 16:59
Message: Comitting final contracts/test fixes.
----
Revision: 329
Author: Skovvart
Date: 2012-05-16 15:45
Message: Bon compilation fixes.
----
Revision: 328
Author: Skovvart
Date: 2012-05-16 15:18
Message: Updated BON, compiled it, fixed some comments and some contracts, removed Printer from BON.
----
Revision: 327
Author: Skovvart
Date: 2012-05-15 17:19
Message: Redid Station formal and informal documentation, added a couple of contracts in Station.cs
----
Revision: 326
Author: Aaes
Date: 2012-05-15 15:51
Message: added comments to all UI classes
----
Revision: 325
Author: Skovvart
Date: 2012-05-15 15:43
Message: Commented StopListening
----
Revision: 324
Author: Skovvart
Date: 2012-05-15 14:47
Message: Should now announce to peers joining after the election has started, that they should start as well
----
Revision: 323
Author: Aaes
Date: 2012-05-15 14:36
Message:
----
Revision: 322
Author: Aaes
Date: 2012-05-15 14:24
Message: when a station is promoted to the manager the amount of stations is checked
----
Revision: 321
Author: Aaes
Date: 2012-05-15 14:16
Message:
----
Revision: 320
Author: Aaes
Date: 2012-05-15 14:06
Message: changed width of shown password on waitingformanagerpage
----
Revision: 319
Author: Aaes
Date: 2012-05-15 14:00
Message:
----
Revision: 318
Author: Skovvart
Date: 2012-05-15 13:51
Message: You might have a manager if you are a manager, when receiving the PublicKeyExchangeCommand reply
----
```

```
Revision: 317
Author: Aaes
Date: 2012-05-15 13:39
Message:
----
Revision: 316
Author: Aaes
Date: 2012-05-15 13:30
Message: dispatcher methods used in notenoughpeers and enoughpeers
----
Revision: 315
Author: Aaes
Date: 2012-05-15 13:20
Message:
----
Revision: 314
Author: Aaes
Date: 2012-05-15 13:19
Message: changed names to danish
----
Revision: 313
Author: Skovvart
Date: 2012-05-15 13:12
Message:
----
Revision: 312
Author: Skovvart
Date: 2012-05-15 13:10
Message: Reorganized some regions in Stations, "implemented" the new IDvlUi features in TestUi
----
Revision: 311
Author: Aaes
Date: 2012-05-15 13:05
Message: NotEnoughPeers and EnoughPeers are implemented
----
Revision: 310
Author: Skovvart
Date: 2012-05-15 12:54
Message: AddPeer and RemovePeer will now announce to the UI when there's enough or not enough peers to
continue the election. The required amount of peers is at the moment set to 1.
----
Revision: 309
Author: Aaes
Date: 2012-05-15 12:53
Message: added enough peers and not enough peers to IDvlUI
----
Revision: 308
Author: Skovvart
Date: 2012-05-15 12:33
Message: Generated passwords now use 10 chars.
----
Revision: 307
Author: Skovvart
Date: 2012-05-15 12:30
Message: Made ShutDownElection and ShutDownElectionCommand notify the UI, renamed default and root
namespace to Aegis_DVL
----
Revision: 306
Author: Aaes
Date: 2012-05-15 12:23
Message: added shutdown UI method
----
Revision: 305
Author: Aaes
Date: 2012-05-14 12:49
Message: changed the window titles
----
Revision: 304
Author: Aaes
Date: 2012-05-14 12:35
Message: changed the icon for the .exe file
----
Revision: 303
Author: Aaes
Date: 2012-05-14 12:31
Message: Changed the icon
----
Revision: 302
Author: Skovvart
Date: 2012-05-11 18:27
Message: Changed PublicKeyExchange failure state requirement to Manager != null
----
Revision: 301
Author: Aaes
Date: 2012-05-11 18:06
```

```
Message:
----
Revision: 300
Author: Aaes
Date: 2012-05-11 17:30
Message:
----
Revision: 299
Author: Aaes
Date: 2012-05-11 17:29
Message:
----
Revision: 298
Author: Aaes
Date: 2012-05-11 17:19
Message:
----
Revision: 297
Author: Aaes
Date: 2012-05-11 16:12
Message:
----
Revision: 296
Author: Skovvart
Date: 2012-05-11 14:28
Message: Disabled contracts, upped the connect time-out
----
Revision: 295
Author: Aaes
Date: 2012-05-11 14:27
Message:
----
Revision: 294
Author: Skovvart
Date: 2012-05-11 13:54
Message: and the same thing again
----
Revision: 293
Author: Skovvart
Date: 2012-05-11 13:51
Message: Will no longer attempt to remove peers that aren't added (when publickeyexchange fails)
----
Revision: 292
Author: Skovvart
Date: 2012-05-11 12:36
Message: Removed some todo-comments.
----
Revision: 291
Author: Skovvart
Date: 2012-05-11 12:06
Message: Final (?) code coverage whoring
----
Revision: 290
Author: Skovvart
Date: 2012-05-10 17:48
Message: Tests fixed and updated to satisfy coverage requirements.
----
Revision: 289
Author: Skovvart
Date: 2012-05-10 16:58
Message: Added revoke when target fails to receive BallotReceived, added cpr, password options in station
----
Revision: 288
Author: Skovvart
Date: 2012-05-10 16:24
Message: Commented and contracted constructors.
----
Revision: 287
Author: Aaes
Date: 2012-05-10 15:13
Message: changed titles on dialogs
----
Revision: 286
Author: Aaes
Date: 2012-05-10 15:11
Message:
----
Revision: 285
Author: Skovvart
Date: 2012-05-10 15:11
Message: Code analysis fixes
----
Revision: 284
Author: Aaes
Date: 2012-05-10 14:41
```

```
Message:
----
Revision: 283
Author: Skovvart
Date: 2012-05-10 14:13
Message: Code coverage requirements completed (for now)
----
Revision: 282
Author: Skovvart
Date: 2012-05-10 12:22
Message: Removed logging from IsAliveCommands since it gave problems when trying to discover peers.
----
Revision: 281
Author: Skovvart
Date: 2012-05-09 18:23
Message: Still problems with the logger and DiscoverNetworkMachines. Threading problem?
----
Revision: 280
Author: Aaes
Date: 2012-05-09 15:40
Message: Added an icon for the program
----
Revision: 279
Author: Skovvart
Date: 2012-05-09 14:22
Message: Now listens to begin with
----
Revision: 278
Author: Aaes
Date: 2012-05-09 14:22
Message:
----
Revision: 277
Author: Skovvart
Date: 2012-05-09 14:20
Message:
----
Revision: 276
Author: Skovvart
Date: 2012-05-09 14:18
Message: ..neither can the communicator
----
Revision: 275
Author: Skovvart
Date: 2012-05-09 14:16
Message: Can't assume that the logger exists when the DB is created
----
Revision: 274
Author: Skovvart
Date: 2012-05-09 14:11
Message: Logging mostly implemented
----
Revision: 273
Author: Skovvart
Date: 2012-05-09 13:24
Message: Updated some tests
----
Revision: 272
Author: Aaes
Date: 2012-05-09 13:17
Message: comments in UIHandler
----
Revision: 271
Author: Skovvart
Date: 2012-05-09 13:02
Message: Some fixes, updated tests, removed some deprecated constructors
----
Revision: 270
Author: Aaes
Date: 2012-05-09 12:27
Message: A PDF file called "Manual" will be opened when the Help->User Manual is pressed, it must be placed
in "UI/bin/Debug" atm
----
Revision: 269
Author: Skovvart
Date: 2012-05-08 16:22
Message: Finally found the bug that caused tests to loop forever.
----
Revision: 268
Author: Aaes
Date: 2012-05-08 15:43
Message: comments in UIHandler
----
Revision: 267
Author: Aaes
```

```
Date: 2012-05-08 14:54
Message: PDF generator changed test data
----
Revision: 266
Author: Skovvart
Date: 2012-05-07 16:45
Message: Fixed some tests, removed Printer (since it probably shouldn't be a part of the solution)
----
Revision: 265
Author: Aaes
Date: 2012-05-07 13:01
Message:
----
Revision: 264
Author: Skovvart
Date: 2012-05-07 12:53
Message: ForEach should not use a local collection, so it wont accidentally be modified during execution.
----
Revision: 263
Author: Aaes
Date: 2012-05-04 16:58
Message:
----
Revision: 262
Author: Skovvart
Date: 2012-05-04 16:55
Message: PromoteNewManager should not update the target UI
----
Revision: 261
Author: Aaes
Date: 2012-05-04 16:48
Message:
----
Revision: 260
Author: Aaes
Date: 2012-05-04 16:27
Message:
----
Revision: 259
Author: Skovvart
Date: 2012-05-04 16:19
Message: ElectNewManager now notifies the UI if the new manager is itself.
----
Revision: 258
Author: Aaes
Date: 2012-05-04 16:19
Message:
----
Revision: 257
Author: Aaes
Date: 2012-05-04 16:18
Message: now a station can become a manager UI-wise
----
Revision: 256
Author: Aaes
Date: 2012-05-04 16:09
Message:
----
Revision: 255
Author: Aaes
Date: 2012-05-04 16:04
Message:
----
Revision: 254
Author: Aaes
Date: 2012-05-04 15:59
Message: markAsConnected should work as intended
----
Revision: 253
Author: Aaes
Date: 2012-05-04 15:51
Message:
----
Revision: 252
Author: Aaes
Date: 2012-05-04 15:50
Message: now only one update thread will be active at a time and it will be aborted when you leave the window
----
Revision: 251
Author: Aaes
Date: 2012-05-04 15:35
Message:
----
Revision: 250
Author: Aaes
```

```
Date: 2012-05-04 15:30
Message:
----
Revision: 249
Author: Aaes
Date: 2012-05-04 15:22
Message: the ballot response dialogs will have a , MessageBoxImage.Stop if the response was false
----
Revision: 248
Author: Aaes
Date: 2012-05-04 15:12
Message: fixed ballotrequestreply to use the right dispatcher
----
Revision: 247
Author: Aaes
Date: 2012-05-04 15:04
Message: Loading bar for updating
----
Revision: 246
Author: Skovvart
Date: 2012-05-04 15:00
Message: CryptoCommand should also use the appropriate key now.
----
Revision: 245
Author: Skovvart
Date: 2012-05-04 14:55
Message: CryptoCommand will now also accept messages from yourself
----
Revision: 244
Author: Skovvart
Date: 2012-05-04 14:36
Message: Removed contract that requires that the Iv setter requires that the value is different, since it
causes problems when sending the message to yourself
----
Revision: 243
Author: Aaes
Date: 2012-05-04 14:27
Message: update label added to overview and manageroverview
----
Revision: 242
Author: Skovvart
Date: 2012-05-04 14:16
Message: Database get no longer throws exception when the voternumber doesn't match the cpr, only returns
BallotStatus.NotAvailable
----
Revision: 241
Author: Aaes
Date: 2012-05-04 14:14
Message: the PopulateList methods should be in seperate threads now
----
Revision: 240
Author: Aaes
Date: 2012-05-04 14:02
Message:
----
Revision: 239
Author: Skovvart
Date: 2012-05-04 14:00
Message: ..fixed yet again
----
Revision: 238
Author: Skovvart
Date: 2012-05-04 13:58
Message: Fixed nullcheck
----
Revision: 237
Author: Skovvart
Date: 2012-05-04 13:56
Message: NewIv should always be different from the old one
----
Revision: 236
Author: Skovvart
Date: 2012-05-04 13:45
Message: When the manager is sending cryptocommands to itself, it will now use the right encryption key
(since the manager isn't found in its peerlist)
----
Revision: 235
Author: Aaes
Date: 2012-05-04 13:29
Message: slight optimization of the populateList() methods
----
Revision: 234
Author: Skovvart
Date: 2012-05-04 13:17
Message: Catching Asn1ParsingException as well
```

```
505  ----
506  Revision: 233
507  Author: Aaes
508  Date: 2012-05-04 13:14
509  Message: inactive peers are removed from the peerlist when the overview and manageroverview lists are updated
510  ----
511  Revision: 232
512  Author: Skovvart
513  Date: 2012-05-04 13:13
514  Message: Reduced and changed IP ranges for ITU...
515  ----
516  Revision: 231
517  Author: Skovvart
518  Date: 2012-05-03 17:02
519  Message: Logger updated to use logName instead of a fixed string
520  ----
521  Revision: 230
522  Author: Aaes
523  Date: 2012-05-03 16:45
524  Message: all menuitems in the file menu are now disabled in the endelectionwindow
525  ----
526  Revision: 229
527  Author: Aaes
528  Date: 2012-05-03 16:40
529  Message: you can now close the application from the files menu if you have the master password but not in
     the EndElectionPage
530  ----
531  Revision: 228
532  Author: Aaes
533  Date: 2012-05-03 16:29
534  Message: a user cannot press OK in the acceptStationDialog, AcceptManagerDialog and
     CheckMasterPasswordDialog unless he has actually written something
535  ----
536  Revision: 227
537  Author: Aaes
538  Date: 2012-05-03 16:24
539  Message: It is now impossible to remove stations you are not already connected to in the ManagerOverviewPage
540  ----
541  Revision: 226
542  Author: Aaes
543  Date: 2012-05-03 16:22
544  Message: the station window will appear in the middle of the screen on open
545  ----
546  Revision: 225
547  Author: Aaes
548  Date: 2012-05-03 16:20
549  Message: AcceptStationDialog, AcceptManagerDialog and checkmasterPasswordialog now focuses thier
     passwordboxes on startup and Esc is bound to cancel and Enter is bound to OK
550  ----
551  Revision: 224
552  Author: Skovvart
553  Date: 2012-05-03 16:20
554  Message: PublicKeyExchangeCommand will keep asking for a new password when unable to get a key from the
     provided. Should stop on cancel.
555  ----
556  Revision: 223
557  Author: Aaes
558  Date: 2012-05-03 16:15
559  Message: translated list headers and enabled cancel on the "gør til manager" button
560  ----
561  Revision: 222
562  Author: Aaes
563  Date: 2012-05-03 16:12
564  Message: AcceptStationDialog and AcceptManagerDialog now use password boxes instead of textboxes
565  ----
566  Revision: 221
567  Author: Aaes
568  Date: 2012-05-03 16:11
569  Message: showPasswordwindow wasnt used anymore and was deleted
570  ----
571  Revision: 220
572  Author: Aaes
573  Date: 2012-05-03 16:10
574  Message: added a filter to the export save file dialog
575  ----
576  Revision: 219
577  Author: Aaes
578  Date: 2012-05-03 16:08
579  Message: wipes the shown password on the manager when a reply is received
580  ----
581  Revision: 218
582  Author: Aaes
583  Date: 2012-05-03 16:02
584  Message:
585  ----
```

```
Revision: 217
Author: Aaes
Date: 2012-05-03 16:02
Message: The passwords shown at connect are not shown in dialogs anymore
----
Revision: 216
Author: Skovvart
Date: 2012-05-03 15:03
Message: EndElectionCommand now notifies UI
----
Revision: 215
Author: Aaes
Date: 2012-05-03 14:59
Message:
----
Revision: 214
Author: Aaes
Date: 2012-05-03 14:59
Message: fixed showpasswordonmanager
----
Revision: 213
Author: Aaes
Date: 2012-05-03 14:47
Message: ElectionStarted and ElectionEnded should now work
----
Revision: 212
Author: Skovvart
Date: 2012-05-03 14:43
Message: Generated password length decreased to 2 for testing purposes.
----
Revision: 211
Author: Aaes
Date: 2012-05-03 14:43
Message: end election cancel works and the ballotRequestPage constructor is fixed
----
Revision: 210
Author: Skovvart
Date: 2012-05-03 14:38
Message: StartElectionCommand notifies the UI
----
Revision: 209
Author: Skovvart
Date: 2012-05-03 14:29
Message: Logger not controls the backup of older logs instead of the UI.
----
Revision: 208
Author: Aaes
Date: 2012-05-03 14:10
Message: The right dispacther now opens a dialog
----
Revision: 207
Author: Skovvart
Date: 2012-05-03 13:54
Message: Updated DiscoverNetworkMachines to use a CountdownEvent.
----
Revision: 206
Author: Skovvart
Date: 2012-05-03 13:26
Message: Fixed and removed some TODO's
----
Revision: 205
Author: Skovvart
Date: 2012-05-03 13:19
Message: DiscoverNetworkMachines now waits for all threads to finish. DiscoverPeers no longer checks that
    StationActive since DiscoverNetworkMachines does this.
----
Revision: 204
Author: Skovvart
Date: 2012-05-03 13:08
Message: Optimized Send slightly
----
Revision: 203
Author: Aaes
Date: 2012-05-03 12:42
Message:
----
Revision: 202
Author: Aaes
Date: 2012-05-03 12:41
Message:
----
Revision: 201
Author: Aaes
Date: 2012-05-03 12:33
Message:
```

```
670  ----
671  Revision: 200
672  Author: Aaes
673  Date: 2012-05-03 12:17
674  Message:
675  ----
676  Revision: 199
677  Author: Skovvart
678  Date: 2012-05-03 00:31
679  Message: Updated cleanup for some tests.
680  ----
681  Revision: 198
682  Author: Skovvart
683  Date: 2012-05-03 00:23
684  Message: Updated some tests (a lot still broken due to lack of a UI), attempted changing Send to split the
     message into multiple packets to better be able to send large messages (Sync command)
685  ----
686  Revision: 197
687  Author: Skovvart
688  Date: 2012-05-02 21:04
689  Message:
690  ----
691  Revision: 196
692  Author: Skovvart
693  Date: 2012-05-02 20:57
694  Message:
695  ----
696  Revision: 195
697  Author: Skovvart
698  Date: 2012-05-02 18:51
699  Message:
700  ----
701  Revision: 194
702  Author: Aaes
703  Date: 2012-05-02 18:51
704  Message:
705  ----
706  Revision: 193
707  Author: Skovvart
708  Date: 2012-05-02 18:45
709  Message:
710  ----
711  Revision: 192
712  Author: Aaes
713  Date: 2012-05-02 18:45
714  Message:
715  ----
716  Revision: 191
717  Author: Skovvart
718  Date: 2012-05-02 18:08
719  Message:
720  ----
721  Revision: 190
722  Author: Skovvart
723  Date: 2012-05-02 18:06
724  Message:
725  ----
726  Revision: 189
727  Author: Aaes
728  Date: 2012-05-02 17:34
729  Message:
730  ----
731  Revision: 188
732  Author: Skovvart
733  Date: 2012-05-02 17:34
734  Message: Bugfixes
735  ----
736  Revision: 187
737  Author: Aaes
738  Date: 2012-05-02 16:03
739  Message: all dialogs should focus and appear in the right position now (middle of the screen)
740  ----
741  Revision: 186
742  Author: Skovvart
743  Date: 2012-05-02 16:03
744  Message: Dispatcher thread handling
745  ----
746  Revision: 185
747  Author: Aaes
748  Date: 2012-05-02 15:43
749  Message:
750  ----
751  Revision: 184
752  Author: Skovvart
753  Date: 2012-05-02 15:43
```

```
Message:
----
Revision: 183
Author: Aaes
Date: 2012-05-02 15:31
Message: now the dialogs showing passwords are not modal anymore
----
Revision: 182
Author: Skovvart
Date: 2012-05-02 15:27
Message:
----
Revision: 181
Author: Skovvart
Date: 2012-05-02 15:16
Message:
----
Revision: 180
Author: Skovvart
Date: 2012-05-02 15:14
Message:
----
Revision: 179
Author: Skovvart
Date: 2012-05-02 15:12
Message: Test code
----
Revision: 178
Author: Aaes
Date: 2012-05-02 15:02
Message: comments in UiHandler
----
Revision: 177
Author: Skovvart
Date: 2012-05-02 15:01
Message: Listener loop added.
----
Revision: 176
Author: Aaes
Date: 2012-05-02 14:46
Message:
----
Revision: 175
Author: Skovvart
Date: 2012-05-02 14:44
Message: New constructors taking IDvlUi
----
Revision: 174
Author: Skovvart
Date: 2012-05-02 14:35
Message: OCD reformatting
----
Revision: 173
Author: Aaes
Date: 2012-05-02 14:33
Message: added commments to IDvlUi
----
Revision: 172
Author: Skovvart
Date: 2012-05-02 13:55
Message: Moved UI interface to proper solution
----
Revision: 171
Author: Skovvart
Date: 2012-05-02 13:53
Message: Moved UI interface to DVL solution, updated placeholder ui-method calls to the interface ones,
    updated some commands to properly notify the UI
----
Revision: 170
Author: Aaes
Date: 2012-05-02 13:52
Message:
----
Revision: 169
Author: Aaes
Date: 2012-05-02 13:50
Message: added a cancel button to the BallotCPRRequestWindow
----
Revision: 168
Author: Aaes
Date: 2012-05-02 13:47
Message:
----
Revision: 167
Author: Aaes
```

```
Date: 2012-05-02 13:46
Message:
----
Revision: 166
Author: Aaes
Date: 2012-05-02 13:46
Message: Now the assorted windows can react to a reply concerning whether or not to hand out a ballot
----
Revision: 165
Author: Skovvart
Date: 2012-05-02 13:27
Message: Made it compilable again
----
Revision: 164
Author: Skovvart
Date: 2012-05-02 13:10
Message:
----
Revision: 163
Author: Skovvart
Date: 2012-05-02 13:10
Message: Updated UI Interface for Dvl purposes
----
Revision: 162
Author: Skovvart
Date: 2012-05-02 12:45
Message: PKExchangeCmd updated to be "UI ready"
----
Revision: 161
Author: Aaes
Date: 2012-05-02 12:45
Message: Now the Done and Only CPR buttons will only be enabled when the sufficient amount of characters
are present
----
Revision: 160
Author: Aaes
Date: 2012-05-02 12:30
Message: added checks for empty CPR and voter number textboxes
----
Revision: 159
Author: Aaes
Date: 2012-05-02 12:24
Message: When you request a ballot there is now a check for whether the voter exists/have voted
----
Revision: 158
Author: Aaes
Date: 2012-05-02 12:19
Message:
----
Revision: 157
Author: Skovvart
Date: 2012-05-01 22:03
Message: Updated most tests to work with the "new" station constructors.
----
Revision: 156
Author: Skovvart
Date: 2012-05-01 17:57
Message: Increased the buffersize, so it can now actually load the data-set from disk.
----
Revision: 155
Author: Skovvart
Date: 2012-05-01 17:49
Message: Updated small dataset (in dropbox), updated EncryptedVoterData to no longer use a tuple (as it was
giving serilization issues), updated UIHandlers import somewhat.
----
Revision: 154
Author: Skovvart
Date: 2012-05-01 16:37
Message: UI now handles Log.sqlite properly
----
Revision: 153
Author: Aaes
Date: 2012-05-01 16:15
Message:
----
Revision: 152
Author: Aaes
Date: 2012-05-01 16:09
Message: the back button on the MasterPassword page now goes to a typechoicepage instead of a dataload page
----
Revision: 151
Author: Aaes
Date: 2012-05-01 16:05
Message: the types password is now dots instead of letters
----
```

```
Revision: 150
Author: Skovvart
Date: 2012-05-01 16:00
Message: Updated Dispose to check that Logger and Crypto are only disposed when set.
----
Revision: 149
Author: Skovvart
Date: 2012-05-01 15:58
Message: ValidMasterPassword checks for null. Rarely relevant, but can cause exception in UI otherwise.
----
Revision: 148
Author: Aaes
Date: 2012-05-01 15:57
Message: When the master password is check the cancel button no longer pops up a prompt saying incorrect
password
----
Revision: 147
Author: Aaes
Date: 2012-05-01 15:53
Message: Check master password dialog changes
----
Revision: 146
Author: Skovvart
Date: 2012-05-01 15:44
Message: Always override database
----
Revision: 145
Author: Aaes
Date: 2012-05-01 15:43
Message: moved a statement
----
Revision: 144
Author: Skovvart
Date: 2012-05-01 15:35
Message:
----
Revision: 143
Author: Aaes
Date: 2012-05-01 15:29
Message: You can now choose "All files" in the dataload page
----
Revision: 142
Author: Skovvart
Date: 2012-05-01 15:29
Message: Added App.Config to allow mixed assembly (though we don't know what mixed assembly this is..
File.Exists?)
----
Revision: 141
Author: Aaes
Date: 2012-05-01 15:28
Message: when a station goes back the station object is disposed
----
Revision: 140
Author: Aaes
Date: 2012-05-01 15:10
Message:
----
Revision: 139
Author: Skovvart
Date: 2012-05-01 14:19
Message: Removed a and updated some constructors. Updated other files where necessary.
----
Revision: 138
Author: Skovvart
Date: 2012-05-01 14:09
Message: Made tests compilable again. Made many tests use "using".
----
Revision: 137
Author: Aaes
Date: 2012-05-01 13:25
Message: The master password is stored from the MasterPasswordPage to the DataLoadPage in order to
correctly initialize the station, it is set to null afterwards
----
Revision: 136
Author: Aaes
Date: 2012-05-01 13:21
Message: comments and finalizing of IUIHandler
----
Revision: 135
Author: Skovvart
Date: 2012-05-01 13:16
Message: "Broke" tests to make them compile, while making the code more ready for release. Fix tests later.
----
Revision: 134
Author: Aaes
```

```
Date: 2012-05-01 13:13
Message:
----
Revision: 133
Author: Skovvart
Date: 2012-05-01 13:07
Message: Made GeneratePassword static (which means it's not part of the interface..)
----
Revision: 132
Author: Aaes
Date: 2012-05-01 13:04
Message: renamed Overview to OverviewPage and rearranged the order of dataloadpage and masterpassword page
----
Revision: 131
Author: Skovvart
Date: 2012-05-01 12:57
Message: AllData calls ToArray() so we don't get WhereSelector which isn't serializable
----
Revision: 130
Author: Aaes
Date: 2012-05-01 12:53
Message: Import and export implemented in the UIHandler
----
Revision: 129
Author: Aaes
Date: 2012-05-01 12:45
Message: Corrected UIHandler to handle new export / import methods
----
Revision: 128
Author: Skovvart
Date: 2012-05-01 12:44
Message: Updated to work with new signature for Import and removed Export
----
Revision: 127
Author: Skovvart
Date: 2012-05-01 12:39
Message: Removed export (data can be exported through AllData), changed Import to just take a dataset
instead of a lambda importing the dataset.
----
Revision: 126
Author: Aaes
Date: 2012-05-01 12:30
Message: Cleaned up UI classes
----
Revision: 125
Author: Skovvart
Date: 2012-05-01 12:20
Message: Made SyncCommand use primitive types, updated test. Potential problem with masterpassword in
constructor in logger and sqlitedb
----
Revision: 124
Author: Aaes
Date: 2012-05-01 12:18
Message: Renamed UiHandler back to UIHandler and remade some missing methods
----
Revision: 123
Author: Aaes
Date: 2012-04-30 16:49
Message: Comments on the UIHandler and in some other UI classes
----
Revision: 122
Author: Skovvart
Date: 2012-04-30 16:43
Message: Fixed some serialization problems with SyncCommand, added a test, added a comment for a missing
parameter in IDatabase, added the option of not creating dummy databases (should be removed soonish
alltogether)
----
Revision: 121
Author: Skovvart
Date: 2012-04-30 15:54
Message: No longer overriding Master.pw
----
Revision: 120
Author: Skovvart
Date: 2012-04-27 11:06
Message: Note added
----
Revision: 119
Author: Skovvart
Date: 2012-04-27 10:25
Message: Added comment. Also, previous update uncommented a couple of things in UiHandler, make sure it
doesn't cause problems.
----
Revision: 118
Author: Skovvart
```

```
Date: 2012-04-27 10:23
Message: Reformatted and renamed some things (Don't hate Aaes :((((()
----
Revision: 117
Author: Skovvart
Date: 2012-04-27 10:12
Message: Added synccommand (untested), updated publickeyexchange (slightly), added bouncycastle to ui
assembly
----
Revision: 116
Author: Aaes
Date: 2012-04-25 15:33
Message: Added a way for every machine to mark a voter only via CPR and masterpassword
----
Revision: 115
Author: Skovvart
Date: 2012-04-25 15:23
Message: [Serializable] på den nye command
----
Revision: 114
Author: Skovvart
Date: 2012-04-25 15:19
Message: RequestBallot (CPR, masterpassword) added to station. Master-password hash is saved to Master.pw
so it persists even when program terminates.
----
Revision: 113
Author: Aaes
Date: 2012-04-24 16:51
Message: To request a ballot using only the CPR the masterpassword is required
----
Revision: 112
Author: Skovvart
Date: 2012-04-24 16:51
Message: Added (CPR, masterPassword) access to the database.
----
Revision: 111
Author: Aaes
Date: 2012-04-24 16:43
Message: It is possible to export the data from the files menu
----
Revision: 110
Author: Aaes
Date: 2012-04-24 16:29
Message: The population of the lists in manageroverviewpage and in overview is optimized
----
Revision: 109
Author: Skovvart
Date: 2012-04-24 16:22
Message: Implemented masterpassword in station (not commands, etc).
----
Revision: 108
Author: Aaes
Date: 2012-04-24 16:04
Message: Moar UI!
----
Revision: 107
Author: Aaes
Date: 2012-04-24 15:05
Message:
----
Revision: 106
Author: Skovvart
Date: 2012-04-24 15:04
Message: Updated communicator to handle failures, updated tests. Added GeneratePassword to (I)Crypto. Added
some #regions to IDisposable.
----
Revision: 105
Author: Aaes
Date: 2012-04-24 14:46
Message: Added a checkMasterPW Dialog and revised the Overview window
----
Revision: 104
Author: Aaes
Date: 2012-04-24 14:09
Message: Corrected the folder structure of the UI
----
Revision: 103
Author: Aaes
Date: 2012-04-24 14:03
Message: Added more code to make the merging of station and the UI smoother
----
Revision: 102
Author: Skovvart
Date: 2012-04-23 16:54
Message: Updated unit tests to include logging.
```

```
----
Replacing : /source/Digital Voter List/Digital Voter List/Logging/LogModel.Designer.cs
Replacing : /source/Digital Voter List/Digital Voter List/Logging/LogModel.edmx
Revision: 101
Author: Aaes
Date: 2012-04-23 15:25
Message: UI updates to match the back end
----
Revision: 100
Author: Aaes
Date: 2012-04-22 23:13
Message: added a master password generator
----
Revision: 99
Author: Skovvart
Date: 2012-04-19 00:58
Message: Fixed election algorithm
----
Revision: 98
Author: Skovvart
Date: 2012-04-18 16:14
Message: Trying to add nunit.framework.dll without the rest of NUnit
----
Revision: 97
Author: Aaes
Date: 2012-04-18 16:12
Message: SQLite DLL's added
----
Revision: 96
Author: Skovvart
Date: 2012-04-18 16:01
Message: Fixed method name in XAML
----
Revision: 95
Author: Skovvart
Date: 2012-04-18 16:00
Message: Implemented IDisposable
----
Revision: 94
Author: Aaes
Date: 2012-04-18 15:59
Message: Added dummy methods to several UI windows and made additional functionality
----
Revision: 93
Author: Skovvart
Date: 2012-04-18 15:07
Message: Updated implemented Logger. Restructered datatypes a bit.
----
Revision: 92
Author: Skovvart
Date: 2012-04-17 16:59
Message: Reformatted logger.
----
Revision: 91
Author: Skovvart
Date: 2012-04-17 16:53
Message: Updated tests, fixed some bugs. 95% code coverage, but still some issues remaining.
----
Revision: 90
Author: Aaes
Date: 2012-04-17 16:44
Message:
----
Revision: 89
Author: Aaes
Date: 2012-04-17 15:20
Message: more UI updates
----
Revision: 88
Author: Aaes
Date: 2012-04-17 13:57
Message: Added navigation between all UI windows and fixed some resizing issues
----
Revision: 87
Author: Aaes
Date: 2012-04-17 13:08
Message: added logging class (not finished) and additional UI windows
----
Revision: 86
Author: Skovvart
Date: 2012-04-16 16:45
Message: Updated tests, changed a lot of command comparisons to use .Equals rather than ==
----
Revision: 85
Author: Aaes
```

```
Date: 2012-04-15 18:28
Message: Added logger and ILogger class + updated the BON to reflect it
----
Revision: 84
Author: Skovvart
Date: 2012-04-12 15:14
Message: Fixed some tests, restructured part of the Database namespace
----
Revision: 83
Author: Aaes
Date: 2012-04-11 15:39
Message: Added PDFGenerator
----
Revision: 82
Author: Skovvart
Date: 2012-04-11 15:35
Message: Removed some TODO's
----
Revision: 81
Author: Skovvart
Date: 2012-04-11 15:11
Message: "Fixed" test
----
Revision: 80
Author: Skovvart
Date: 2012-04-11 15:09
Message: Tests updated, some equality checking fixed.
----
Revision: 79
Author: Skovvart
Date: 2012-04-11 13:57
Message: Updated a couple of classes based on code analysis
----
Revision: 78
Author: Skovvart
Date: 2012-04-11 13:22
Message: Removed Pkcs1v5 padding from Rsa as it makes encryptions incomparable. "Padding" inputbytes with a
{1} to not lose leading zeros.
----
Revision: 77
Author: Skovvart
Date: 2012-04-10 16:45
Message: Crypto doesn't work after all, it seems - same input and key do not generate the same output
----
Replacing : /source/Digital Voter List/Digital Voter List/Database/VoterModel.Designer.cs
Replacing : /source/Digital Voter List/Digital Voter List/Database/VoterModel.edmx
Revision: 76
Author: Skovvart
Date: 2012-04-10 15:42
Message: Didn't get commited for some reason?
----
Revision: 75
Author: Skovvart
Date: 2012-04-10 15:39
Message: Initial implementation of SQLite added.
----
Revision: 74
Author: Skovvart
Date: 2012-04-10 14:02
Message: Renamed some namespaces, changed (I)Communicator to include DiscoverNetworkMachines and
IsListening, removed ValidMessage (since CryptoCommand handles that logic)
----
Revision: 73
Author: Skovvart
Date: 2012-04-09 18:27
Message: todo-comment added
----
Revision: 72
Author: Skovvart
Date: 2012-04-05 19:37
Message: dotCover problem fixed by switching Test project compilation to x86 instead of AnyCPU. Added
temporary PublicKeyExchangedCommand, implemented StationActive in Station, added/fixed a couple of tests.
Made station load an encryptionkey from disk (located in bin directory for now). Made Message serialiable.
----
Revision: 71
Author: Skovvart
Date: 2012-04-03 15:41
Message: Optimized references.
----
Revision: 70
Author: Skovvart
Date: 2012-04-03 15:33
Message: Updated bytetests to dispose of a memorystream. Attempted to figure out what dotCover's problem
is, but to no avail so far.
----
```

```
Revision: 69
Author: Skovvart
Date: 2012-04-03 15:11
Message: Updated a couple of tests and removed unnecessary files from the root source folder.
----
Revision: 68
Author: Skovvart
Date: 2012-04-03 14:55
Message: Added padding to asymmetric encryption, distinguishing between symmetric and asymmetric keys,
updated a few tests and a lot of commands, updated communicator slightly,
----
Revision: 67
Author: Skovvart
Date: 2012-04-03 13:02
Message: Implemented most commands.
----
Revision: 66
Author: Skovvart
Date: 2012-04-02 16:42
Message: Small updates to some tests
----
Revision: 65
Author: Skovvart
Date: 2012-04-02 16:24
Message: Cryptotests added. Some are not passing.
----
Revision: 64
Author: Skovvart
Date: 2012-04-02 15:56
Message: Updated with latest tests.
----
Revision: 63
Author: Skovvart
Date: 2012-04-02 14:33
Message: Initial layout of tests added
----
Revision: 62
Author: Skovvart
Date: 2012-04-02 14:19
Message: Fixed type error in station, worked a bit on communicator tests
----
Revision: 61
Author: Skovvart
Date: 2012-04-02 13:55
Message: Communicator made public, added regions to station, started unit tests
----
Revision: 60
Author: Skovvart
Date: 2012-04-02 13:44
Message:
----
Revision: 59
Author: Aaes
Date: 2012-04-02 12:53
Message:
----
Revision: 58
Author: Skovvart
Date: 2012-04-02 12:52
Message: Small updates
----
Revision: 57
Author: Skovvart
Date: 2012-04-02 12:50
Message: Updated (I)Communicator
----
Revision: 56
Author: Aaes
Date: 2012-04-02 12:50
Message:
----
Revision: 55
Author: Aaes
Date: 2012-04-02 12:50
Message: included a getParent()
----
Revision: 54
Author: Skovvart
Date: 2012-04-02 12:27
Message: compilebon.txt updated to new names
----
Revision: 53
Author: Aaes
Date: 2012-04-01 22:55
Message:
```

```
----
Revision: 52
Author: Aaes
Date: 2012-04-01 22:54
Message:
----
Revision: 51
Author: Aaes
Date: 2012-04-01 22:53
Message: new UI images (concepts)
----
Revision: 50
Author: Skovvart
Date: 2012-03-29 15:06
Message: Most of station done. Need to update BON still.
----
Revision: 49
Author: Skovvart
Date: 2012-03-28 16:19
Message: Updated Message and BON to include an IV
----
Revision: 48
Author: Skovvart
Date: 2012-03-28 15:56
Message: Recompiled BON
----
Revision: 47
Author: Skovvart
Date: 2012-03-28 15:54
Message: Readding commands to solution after conflict.
----
Revision: 46
Author: Skovvart
Date: 2012-03-28 15:51
Message: Added Commands folder and ICommand. Updated BON very very slightly.
----
Revision: 45
Author: Aaes
Date: 2012-03-28 15:46
Message: added BON documentation to the Communicator
----
Revision: 44
Author: Skovvart
Date: 2012-03-28 15:41
Message: Updated compiled bon, added bon compile commands in compilebon.txt
----
Revision: 43
Author: Skovvart
Date: 2012-03-28 15:25
Message: Updated compiled BON
----
Revision: 42
Author: Aaes
Date: 2012-03-28 15:17
Message: added a method to discover all the machines connected to the same network in the workgroup WORKGROUP
----
Revision: 41
Author: Skovvart
Date: 2012-03-28 15:10
Message: Crypto documentation updated
----
Revision: 40
Author: Skovvart
Date: 2012-03-28 14:59
Message: Updated Crypto and ICrypto to "require" the use of initilization vectors for symmetric encryption.
Updated documentation to come.
----
Revision: 39
Author: Skovvart
Date: 2012-03-28 14:44
Message: Crypto mostly implemented, with some TODO notes in comments.
----
Revision: 38
Author: Skovvart
Date: 2012-03-28 14:06
Message: Commented and renamed some utility classes.
----
Revision: 37
Author: Skovvart
Date: 2012-03-28 13:50
Message: Crypto documentation updated
----
Revision: 36
Author: Skovvart
Date: 2012-03-28 13:46
```

```
Message: ICrypto updated
----
Revision: 35
Author: Skovvart
Date: 2012-03-28 13:09
Message: Added utility functions
----
Revision: 34
Author: Skovvart
Date: 2012-03-28 13:04
Message: Skovvart's OCD was satisfied.
----
Revision: 33
Author: Skovvart
Date: 2012-03-28 12:46
Message: Finished IDatabase BON implementation
----
Revision: 32
Author: Aaes
Date: 2012-03-27 16:46
Message: Core.cs now have contracts and BON documentation
----
Revision: 31
Author: Skovvart
Date: 2012-03-27 16:18
Message: IDatabase "done" for now, needs to add the invariant when Station is more complete.
----
Revision: 30
Author: Skovvart
Date: 2012-03-27 16:04
Message: Recompiled information documentation
----
Revision: 29
Author: Skovvart
Date: 2012-03-27 15:50
Message: Re-ignoring .suo
----
Revision: 28
Author: Skovvart
Date: 2012-03-27 15:50
Message: Updated BouncyCastle.Crypto.dll reference
----
Revision: 27
Author: Aaes
Date: 2012-03-27 15:45
Message:
----
Revision: 26
Author: Skovvart
Date: 2012-03-27 15:44
Message: IDatabase formatted, BON slightly updated.
----
Revision: 25
Author: Aaes
Date: 2012-03-27 15:37
Message: wrongly names vars in the BON
----
Revision: 24
Author: Aaes
Date: 2012-03-27 15:37
Message: added documentation and BON methods to the ICrypto class (the old ones are still in there)
----
Revision: 23
Author: Aaes
Date: 2012-03-27 15:20
Message: added documentation and contracts for IPrinter and IScanner
----
Revision: 22
Author: Aaes
Date: 2012-03-27 14:19
Message: second edition of formal BON
----
Revision: 21
Author: Aaes
Date: 2012-03-27 14:14
Message: first edition of formal BON
----
Revision: 20
Author: Skovvart
Date: 2012-03-27 12:29
Message: Fixed Command_Formal.bon
----
Revision: 19
Author: Aaes
Date: 2012-03-27 12:22
```

```
Message: camelcase OCD
----
Revision: 18
Author: Aaes
Date: 2012-03-27 12:21
Message: camelcase OCD
----
Revision: 17
Author: Aaes
Date: 2012-03-27 12:19
Message: Command_formal.bon done
----
Revision: 16
Author: Skovvart
Date: 2012-03-27 12:09
Message: Removed readme.textile
----
Revision: 15
Author: Aaes
Date: 2012-03-27 12:08
Message: Formal BON files added
----
Revision: 14
Author: Skovvart
Date: 2012-03-26 15:59
Message: Recompiled BON documentation
----
Revision: 13
Author: Aaes
Date: 2012-03-26 15:58
Message: Additional revisions to the informal BON
----
Revision: 12
Author: Skovvart
Date: 2012-03-26 14:11
Message: Updated compiled BON documentation
----
Revision: 11
Author: Skovvart
Date: 2012-03-26 14:08
Message: ???
----
Revision: 10
Author: Aaes
Date: 2012-03-26 14:05
Message: more informal BON changes
----
Revision: 9
Author: Skovvart
Date: 2012-03-26 14:01
Message: Updated compiled BON
----
Revision: 8
Author: Skovvart
Date: 2012-03-26 13:58
Message: Removed triplet - tuple of three should do.
----
Revision: 7
Author: Aaes
Date: 2012-03-26 13:52
Message: Changed informal BON - Communicator_Informal.bon
----
Revision: 6
Author: Skovvart
Date: 2012-03-26 13:38
Message: Documentation added
----
Revision: 5
Author: Skovvart
Date: 2012-03-26 13:34
Message: Hello Aaes
----
Revision: 4
Author: Skovvart
Date: 2012-03-26 13:33
Message: Test commit
----
Revision: 3
Author: Skovvart
Date: 2012-03-26 13:14
Message:
----
Revision: 2
Author: Skovvart
Date: 2012-03-26 12:58
```

```
1668  Message: Initial source commit
1669  ----
1670  Revision: 1
1671  Author: www-data
1672  Date: 2011-09-18 15:27
1673  Message: Automatically created readme.textile and /trunk directory. We recommend you to put all your code
      there.
```

17.7 BON

```
1  system_chart DVL
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "An open source digital voter list that keeps track of
   who has been handed a ballot at an election, with a focus on security."
5    cluster DIGITALVOTERLIST description "The various elements of the
   digital voter list system."
6    cluster COREDATATYPES description "Core datatypes used by the digital
   voter list system."
7  end
8
9  cluster_chart DIGITALVOTERLIST
10   indexing
11     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
12   explanation "The various elements of the digital voter list system."
13   class STATION description "A station is a client-machine that
   communicates with its manager, and provides a graphical user interface
   for voters to use when requesting a ballot. A station can also be the
   manager. A manager manages the various stations, and handles
   synchronization of the data. It also has elevated rights compared to a
   station, and can for example manually mark a voter as having been
   handed a ballot (in case he lost his voter card, or the like)."
14   class SCANNER description "A scanner can read a physical voter card
   and extract required information from it."
15   class COMMUNICATOR description "A communicator is responsible for
   securely passing commands between two parties."
16   class CRYPTO description "Crypto is responsible for cryptographic
   functions such as public-key encryption."
17   class DATABASE description "The database-layer is responsible for
   communicating with the database (create, read, update, write). It can
   also perform batch-operations such as importing and exporting the
   database."
18   class COMMAND description "A command is sent over the network and can
   be executed at the destination."
19   class LOGGER description "A log is used to track events in the system."
20   class UI description "A UI is used to interact with human beings. The
   UI must be able to support requirements to be able to interact with the
   Digital Voter List system."
21 end
22
23 cluster_chart COREDATATYPES
24   indexing
25     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
26   explanation "Core datatypes used by the digital voter list system."
27   class CIPHERTEXT description "CipherText is encrypted data."
28   class ASYMMETRICKEY description "An asymmetric key can be used for
   either encryption or decryption of data."
29   class SYMMETRICKEY description "A symmetric key can be used for
   either encryption or decryption of data."
30   class MESSAGE description "A message contains ciphertext of a
   symmetric key, a message encrypted with the symmetric key and a hash
   encrypted with the senders public key. Used for secure communication."
31   class CPR description "A CPR-number is a number identifying a danish
   citizen, consisting of the birthdate and a unique identifier."
32   class VOTERNUMBER description "A voternumber is a unique number used
   in conjunction with the CPR-number to request a ballot."
33   class BALLOTSTATUS description "A ballot status is used in
   conjunction with a cpr-number and a voternumber, and indicates wheither
   status that indicates whether the ballot has been handed out, not
   handed out, or if it is unavailable at the given election venue."
34   class ENCRYPTEDVOTERDATA description "Encrypted voterdata is the
   encrypted combination of CPR, VOTERNUMBER and BALLOTSTATUS."
```

```
35    class LOGENTRY description "A log entry is an entry in a log. It
      contains a message, a time and a level indicating its type."
36    end
```

```
1  class_chart COMMAND
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A command is sent over the network and can be executed
   at the destination."
5  query
6    "Who sent this command?"
7  command
8    "Execute this command!"
9  end
```

```
1  class_chart COMMUNICATOR
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A communicator is responsible for securely passing
   commands between two parties."
5  query
6    "May I have a new communicator with this station as the parent?",
7    "Is this machine listening on this port?",
8    "Who is my parent?",
9    "What are the addresses of machines in the local network?"
10 command
11   "Send this command securely to this target!",
12   "Receive and handle all commands!"
13 constraint
14   "All commands should be secure"
15 end
```

```
1  class_chart CIPHERTEXT
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "CipherText is encrypted data."
5  query
6    "What does this CipherText look like?"
7  constraint
8    "The value of the ciphertext must always be non-void."
9  end
10
11 class_chart ASYMMETRICKEY
12   indexing
13     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
14   explanation "An asymmetric key can be used for either encryption or
   decryption of data."
15 query
16   "What does this asymmetric key look like?"
17 constraint
18   "The value of an asymmetric key must always be non-void."
19 end
20
21 class_chart SYMMETRICKEY
22   indexing
23     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
24   explanation "A symmetric key can be used for either encryption or
   decryption of data."
25 query
26   "What does this symmetric key look like?"
27 constraint
28   "The value of a symmetric key must always be non-void."
29 end
30
31 class_chart MESSAGE
32   indexing
33     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
34   explanation "A message contains the ciphertexts of a symmetric key, a
   command encrypted with the symmetric key and a hash encrypted with the
   senders public key. Used for secure communication."
35 query
36   "What is the initialization vector used to encrypt the command?",
37   "What is the CipherText of the symmetric key used to encrypt the
   command?",
38   "What is the CipherText of the encrypted command?",
39   "What is the CipherText of the senderhash of the command?"
40 end
41
42 class_chart CPR
43   indexing
44     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
45   explanation "A CPR-number is a number identifying a danish citizen,
   consisting of the birthdate and a number."
46 query
47   "What does this CPR-number look like?"
48 constraint
49   "The numeric value of a CPR-number is always greater than zero."
50 end
51
52 class_chart VOTERNUMBER
53   indexing
54     author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
55   explanation "A voternumber is a unique number used in conjunction
   with the CPR-number to request a ballot."
```

```
56  query
57    "What does this voter-number look like?"
58  end
59
60  class_chart BALLOTSTATUS
61    indexing
62      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
63    explanation "A ballot status is used in conjunction with a cpr-number
    and a voternumber, and indicates wheither status that indicates whether
    the ballot has been handed out, not handed out, or if it is unavailable
    at the given election venue."
64  query
65    "What is the status of this ballot?"
66  constraint
67    "A ballot status is always either 'handed out', 'not handed out' or
    'not available'."
68  end
69
70  class_chart ENCRYPTEDVOTERDATA
71    indexing
72      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
73    explanation "Encrypted voterdata is the encrypted combination of CPR,
    VOTERNUMBER and BALLOTSTATUS."
74  query
75    "What is the encrypted CPR-number of this encrypted voterdata?",
76    "What is the encrypted voter-number of this encrypted voterdata?",
77    "What is the encrypted ballot status of this encrypted voterdata?"
78  constraint
79    "All the data must have a value, that is, be non-void."
80  end
81
82  class_chart LOGENTRY
83    indexing
84      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nsbk@itu.dk)"
85    explanation "A log entry is an entry in a log. It contains a message,
    a time and a level indicating its type."
86  query
87    "What is the message of the log entry?",
88    "What type of log entry is this?",
89    "At what time was the log entry added?"
90  constraint
91    "None of the values must be void."
92  end
```

```
1  class_chart CRYPTO
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "Crypto is responsible for cryptographic functions such
   as public-key encryption."
5  query
6    "What is the asymmetric key used for encrypting voterdata at this
   election venue?",
7    "What are the keys for my public key infrastructure?",
8    "What does this look like when it's symmetrically encrypted with this
   key?",
9    "What does this look like when it's symmetrically decrypted with this
   key?",
10   "What is the current initilization vector?",
11   "What does this look like when it's asymmetrically encrypted with
   this key?",
12   "What does this look like when it's asymmetrically decrypted with
   this key?",
13   "What is the hashed value of this?",
14   "May I have a new randomly generated symmetric key?"
15 command
16   "The initilization vector is this!",
17   "Generate a new initilization vector to be used for symmetric
   encryption!"
18 end
```

```
1  class_chart DATABASE
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "The database-layer is responsible for communicating with
   the database (create, read, update, write). It can also perform
   batch-operations such as importing and exporting the database."
5  query
6    "Has this voter received a ballot?",
7    "What does the entire database look like?",
8    "Who is my parent station?"
9  command
10   "This user has received a ballot!",
11   "This user's ballot has been revoked!",
12   "Import this encrypted data into the database!"
13 constraint
14   "After the election has started, the number of rows should never
   change."
15 end
```

```
1  class_chart LOGGER
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A log is used to track events in the system."
5  query
6    "What does the entire log look like?"
7  command
8    "Log this message!"
9  end
```

```
1  class_chart SCANNER
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A scanner can read a physical voter card and extract
   required information from it."
5  query
6    "What is the voter number from this voter card?",
7  constraint
8    "Failure to read the voter card should result in an error."
9  end
```

```
1  class_chart STATION
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A station is a client-machine that communicates with its
     manager, and provides a graphical user interface for voters to use when
     requesting a ballot. A station can also be the manager. A manager
     manages the various stations, and handles synchronization of the data.
     It also has elevated rights compared to a station, and can for example
     manually mark a voter as having been handed a ballot (in case he lost
     his voter card, or the like)."
5  query
6    "What is my address?",
7    "Who is the manager?",
8    "Is there enough active stations in the group to continue operations?",
9    "What is the status of the election?",
10   "Who are my peers?",
11   "How can I manipulate my database?",
12   "How can I communicate with my group?",
13   "How can I encrypt messages?",
14   "How can I log messages?",
15   "How can the user interact with me?",
16   "Am I the manager?",
17   "What is the master password?",
18   "Is this station active?",
19   "What machines on the network respond that they have the digital
     voter list software running?",
20   "Is this string the masterpassword?",
21   "Can I have a new Station that is the manager?",
22   "Can I have a new Station?"
23 command
24   "This station is now the manager!",
25   "This is how you encrypt messages!",
26   "This is how you log messages!",
27   "The master password is this!",
28   "The system is compromised, notify everyone and shut down the
     election!",
29   "Exchange public keys with this machine!",
30   "Start listening to other stations!",
31   "Stop listening to other stations!",
32   "Start the election!",
33   "Add this station to the group!",
34   "Remove this station from the group!",
35   "Start election of a new manager!",
36   "Elect a new manager!",
37   "Request a ballot for this voter!",
38   "This voter received a ballot!",
39   "Revoke this ballot!",
40   "Tell the group to remove this station as a peer!",
41   "Make this station the new manager!",
42   "Announce to all stations that the election has started!",
43   "Announce to all stations that the election has ended!",
44   "Announce to all that they should revoke this update!"
45 constraint
46   "The master password must not be set to null, and the master password
     must not be changed once it's set.",
47   "All addresses must be well-formed, that is, not null.",
48   "When exchanging public keys with a station, the station must be
     active.",
49   "You can not start or stop listening unless you're in the opposite
     state.",
50   "You can not start or end an election unless you're in the opposite
     state.",
```

```
51    "You can not add or remove a peer unless it's either not in or in the
      peer-list.",
52    "The manager must not be active when attempting to elect a new
      manager.",
53    "You can not request a ballot for a voter that has already received a
      ballot, or who can not be found in the database.",
54    "You can not revoke a ballot for a voter that has not received ab
      allot, or who can not be found in the database.",
55    "To announce the adding or removing of peers, to announce that a
      ballot has been received or revoked, to announce the start or end the
      election or to promote a new manager, you must be the manager.",
56    "The address must never be null, nor must the Peer-list."
57  end
```

```
1  class_chart UI
2    indexing
3      author: "Nikolaj Aaes (niaa@itu.dk) & Nicolai Skovvart (nbsk@itu.dk)"
4    explanation "A UI is used to interact with human beings. The UI must
     be able to support requirements to be able to interact with the Digital
     Voter List system."
5  query
6    "What is the key the user typed in to respond to the manager
     initiating a key-exchange?",
7    "What is the password the user typed in when a station is replying to
     a key-exchange?"
8  command
9    "Show this password on the manager machine!",
10   "Show this password on a station machine!",
11   "Let the UI know whether or not the voter can receive a ballot!",
12   "Let the UI know that the election has ended!",
13   "Let the UI know that the election has started!",
14   "Let the UI know that this machine is now the manager!",
15   "Let the UI know that it needs to shut down!",
16   "Let the UI know that there are not enough peers to continue
     execution!",
17   "Let the UI know that there are enough peers to continue execution!"
18 end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster COMMAND
4    component
5      deferred class ICOMMAND
6        feature
7          deferred Sender : IPADDRESS
8            ensure result /= void
9          end
10
11         deferred Execute : void
12           -> s : STATION
13           require s /= void
14         end
15     end
16   end
17 end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster COMMUNICATOR
4    component
5      deferred class ICOMMUNICATOR
6        feature
7
8          deferred IsListening : BOOLEAN
9            -> a : IPADDRESS
10           require a /= void
11         end
12
13         deferred DiscoverNetworkMachines : SEQUENCE[IPADDRESS]
14           ensure result /= void
15         end
16
17         deferred Send : void
18           -> c : COMMAND
19           -> target : IPADDRESS
20           require c /= void and target /= void
21         end
22
23         deferred ReceiveAndHandle : void
24
25         deferred Parent : STATION
26           ensure result /= null
27         end
28
29     end
30   end
31 end
```

```
static_diagram DIGITALVOTERLIST
component
  cluster CORE_DATA_TYPES
  component
    class CIPHERTEXT
      feature
        Value : BYTEARRAY
      invariant
        Value /= void
    end

    class ASYMMETRICKEY
      feature
        Value : ASYMMETRICKEYPARAMETER
      invariant
        Value /= void
    end

    class SYMMETRICKEY
      feature
        Value : BYTEARRAY
      invariant
        Value /= void
    end

    class MESSAGE
      feature
        Iv : BYTEARRAY

        SymmetricKey : CIPHERTEXT

        Command : CIPHERTEXT

        SenderHash : CIPHERTEXT
      invariant
        Iv /= void and SymmetricKey /= void and Command /= void and
SenderHash /= void
    end

    class VOTERNUMBER
      feature
        Value : INTEGER
          ensure
            result /= void and
            result >= 0
        end
    end

   class CPR
      feature
        Value : INTEGER
          ensure
            result /= void and
            result > 0
        end
    end

    class BALLOTSTATUS
      feature
        Value : INTEGER
          ensure
            result /= void and
```

```
62              result >= 0 and
63              result < 3
64              -- 0 is handed out, 1 is not handed out and 2 is not
    available (this would ideally be an ENUM)
65          end
66      end
67
68      class ENCRYPTEDVOTERDATA
69        feature
70          cpr : CIPHERTEXT
71
72          voterNumber : CIPHERTEXT
73
74          ballotStatus : CIPHERTEXT
75        invariant
76          cpr /= void and voterNumber /= void and ballotStatus /= void
77      end
78
79      class LOGENTRY
80        feature
81          Message : VALUE
82          Level : VALUE
83          TimeStamp : INTEGER
84        invariant
85          Message /= void and Level /= void and Timestamp /= void
86      end
87    end
88  end
```

```
static_diagram DIGITALVOTERLIST
component
  cluster CRYPTO
  component
    deferred class ICRYPTO
      feature

      VoterDataEncryptionKey : ASYMMETRICKEY
        ensure result /= void
      end

      SetVoterDataEncryptionKey : void
        -> key : ASYMMETRICKEY

      Keys : SET[ASYMMETRICKEY]
        ensure result /= void
      end

      AsymmetricDecrypt : BYTEARRAY
        -> c : CIPHERTEXT
        -> k : ASYMMETRICKEY
        require
          c /= void and
          k /= void
        ensure result /= void
      end

      AsymmetricEncrypt : CIPHERTEXT
        -> b : BYTEARRAY
        -> k : ASYMMETRICKEY
        require
          b /= void and
          k /= void
        ensure result /= void
      end

      SymmetricDecrypt : BYTEARRAY
        -> c : CIPHERTEXT
        -> k : SYMMETRICKEY
        require
          c /= void and
          k /= void
        ensure result /= void
      end

      SymmetricEncrypt : CIPHERTEXT
        -> b : BYTEARRAY
        -> k : SYMMETRICKEY
        require
          b /= void and
          k /= void
        ensure result /= void
      end

      Hash : BYTEARRAY
        -> b : BYTEARRAY
        require b /= void
      end

      SetIv : void
        -> b : BYTEARRAY
        require b /= void
```

```
63            ensure GetIv = b
64          end
65
66          GetIv : BYTEARRAY
67            ensure result /= void
68          end
69
70          NewIv : void
71            ensure GetIv /= void and GetIv /= old getIv
72          end
73
74          GenerateSymmetricKey : BYTEARRAY
75            ensure result /= void
76          end
77        end
78      end
79    end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster DATABASE
4    component
5      deferred class IDATABASE
6        feature
7
8          deferred GetBallotStatus : BALLOTSTATUS
9            -> vn : VOTERNUMBER
10           -> cpr : CPR
11           require
12             vn /= void and cpr /= void
13         end
14
15         deferred SetBallotStatus : void
16           -> vn : VOTERNUMBER
17           -> cpr : CPR
18           -> bs : BALLOTSTATUS
19           require
20             GetBallotStatus(vn, cpr) /= BALLOTSTATUS.Unavailable and bs
   /= BALLOTSTATUS.Unavailable and ((GetBallotStatus(vn, cpr) =
   BALLOTSTATUS.NotReceived and bs = BallotStatus.Received) or
   (GetBallotStatus(vn, cpr) = BALLOTSTATUS.Received and bs =
   BallotStatus.NotReceived))
21           ensure GetBallotstatus(vn, cpr) = bs
22         end
23
24         deferred GetBallotStatusCPROnly : BALLOTSTATUS
25           -> cpr : CPR
26           -> pswd : STRING
27           require
28             pswd /= void and Parent.ValidMasterPassword(pswd)
29         end
30
31         deferred SetBallotStatusCPROnly : void
32           -> cpr : CPR
33           -> bs : BALLOTSTATUS
34           -> pswd : STRING
35           require
36             pswd /= void and Parent.ValidMasterPassword(pswd) and
   GetBallotStatusCPROnly(cpr, pswd) /= BALLOTSTATUS.Unavailable and bs /=
   BALLOTSTATUS.Unavailable and ((GetBallotStatusCPROnly(cpr, pswd) =
   BALLOTSTATUS.NotReceived and bs = BallotStatus.Received) or
   (GetBallotStatusCPROnly(cpr, pswd) = BALLOTSTATUS.Received and bs =
   BallotStatus.NotReceived))
37           ensure GetBallotstatusCPROnly(cpr, pswd) = bs
38         end
39
40         deferred AllData : SEQUENCE[ENCRYPTEDVOTERDATA]
41           ensure result /= void
42         end
43
44         deferred Parent : STATION
45           ensure result /= void
46         end
47
48         deferred Import : void
49           -> data : SEQUENCE[ENCRYPTEDVOTERDATA]
50           require data /= void
51         end
52     end
53   end
```

```
54  end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster LOGGER
4    component
5      deferred class ILOGGER
6        feature
7
8          deferred Log : void
9            -> message : VALUE
10           -> level : VALUE
11           require message /= void
12         end
13
14         deferred Export : SEQUENCE[LOGENTRY]
15           ensure result /= void
16         end
17     end
18   end
19 end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster SCANNER
4    component
5      deferred class ISCANNER
6        feature
7
8          deferred scan : VOTERNUMBER
9
10     end
11   end
12 end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster STATIONANDMANAGER
4    component
5      deferred class STATION
6        feature
7          Address : IPADDRESS
8
9          Manager : IPADDRESS
10
11         SetManager : void
12           -> address : IPADDRESS
13           require address /= void
14           ensure Manager = address
15         end
16
17         EnoughStations : BOOLEAN
18
19         ElectionInProgress : BOOLEAN
20
21         Peers : SORTED_LIST[IPADDRESS]
22
23         Database : IDATABASE
24
25         Communicator : ICOMMUNICATOR
26
27         Crypto : ICRYPTO
28
29         SetCrypto : void
30           -> newcrypto : ICRYPTO
31           require newcrypto /= void
32           ensure Crypto = newcrypto
33         end
34
35         Logger : ILOGGER
36
37         SetLogger : void
38           -> newlogger : ILOGGER
39           require newlogger /= void
40           ensure Logger = newlogger
41         end
42
43         UI : IDVLUI
44
45         IsManager : BOOLEAN
46
47         Listening : BOOLEAN
48
49         MasterPassword : VALUE
50
51         SetMasterPassword : void
52           -> password : VALUE
53           require password /= void and MasterPassword = void
54           ensure MasterPassword = password
55         end
56
57         StationActive : BOOLEAN
58           -> address : IPADDRESS
59           require address /= void
60         end
61
62         DiscoverPeers : SEQUENCE[IPADDRESS]
```

```
63            ensure result /= void
64          end
65
66          ValidMasterPassword : BOOLEAN
67            -> password : STRING
68            require password /= void
69          end
70
71          ShutDownElection : void
72
73          ExchangePublicKeys : void
74            -> address : IPADDRESS
75            require address /= void and StationActive(address)
76          end
77
78          StartListening : void
79            require not Listening
80            ensure Listening
81          end
82
83          StopListening : void
84            require Listening
85            ensure not Listening
86          end
87
88          StartElection : void
89            require not ElectionInProgress
90            ensure ElectionInProgress
91          end
92
93          EndElection : void
94            require ElectionInProgress
95            ensure not ElectionInProgress
96          end
97
98          AddPeer : void
99            -> address : IPADDRESS
100           -> key : ASYMMETRICKEY
101           require address /= void and not Peers.Contains(address)
102           ensure Peers.Contains(address)
103         end
104
105         RemovePeer : void
106           -> address : IPADDRESS
107           require address /= void and Peers.Contains(address)
108           ensure not Peers.Contains(address)
109         end
110
111         StartNewManagerElection : void
112
113         ElectNewManager : void
114           require not StationActive(Manager)
115           ensure Manager /= old Manager
116         end
117
118         RequestBallot : void
119           -> voterNumber : VOTERNUMBER
120           -> cpr : CPR
121           require Database.get(voterNumber, cpr) = NOTRECEIVED
122         end
123
124         RequestBallotCPROnly : void
```

```
125             -> cpr : CPR
126             -> password : STRING
127             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = NOTRECEIVED
128           end
129
130           BallotReceived : void
131             -> voterNumber : VOTERNUMBER
132             -> cpr : CPR
133             require Database.get(voterNumber, cpr) = NOTRECEIVED
134             ensure Database.get(voterNumber, cpr) = RECEIVED
135           end
136
137           BallotReceivedCPROnly : void
138             -> cpr : CPR
139             -> password : STRING
140             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = NOTRECEIVED
141             ensure Database.get(cpr, password) = RECEIVED
142           end
143
144           RevokeBallot : void
145             -> voterNumber : VOTERNUMBER
146             -> cpr : CPR
147             require Database.get(voterNumber, cpr) = RECEIVED
148             ensure Database.get(voterNumber, cpr) = NOTRECEIVED
149           end
150
151           RevokeBallotCPROnly : void
152             -> cpr : CPR
153             -> password : STRING
154             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = RECEIVED
155             ensure Database.get(cpr, password) = NOTRECEIVED
156           end
157
158           AnnounceAddPeer : void
159             -> newPeerAddress : IPADDRESS
160             -> newPeerKey : ASYMMETRICKEY
161             require IsManager and newPeerAddres /= void
162           end
163
164           AnnounceRemovePeer : void
165             -> removePeerAddress : IPADDRESS
166             require IsManager and removePeerAddress /= void
167           end
168
169           PromoteNewManager : void
170             -> newManagerAddress : IPAddress
171             require IsManager and newManagerAddress /= void
172           end
173
174           AnnounceStartElection : void
175             require IsManager and not ElectionInProgress
176             ensure ElectionInProgress
177           end
178
179           AnnounceEndElection : void
180             require IsManager and ElectionInProgress
181             ensure not ElectionInProgress
182           end
183
```

```
184           invariant
185             Address /= void and Peers /= void
186       end
187     end
188   end
```

```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster UI
4    component
5      deferred class IUI
6        feature
7          ManagerExchangingKey : STRING
8            -> ip : IPADDRESS
9          StationExchangingKey : STRING
10           -> ip : IPADDRESS
11         ShowPasswordOnManager : void
12           -> pswd : STRING
13         ShowPasswordOnStation : void
14           -> pswd : STRING
15         BallotRequestReply : void
16           -> handOutBallot : BOOLEAN
17         ElectionEnded : void
18         ElectionStarted : void
19         IsNowManager : void
20         ShutDown : void
21         NotEnoughPeers : void
22         EnoughPeers : void
23     end
24   end
25 end
```